Escuela Politécnica Superior

# Aplicación de FIWARE en un destino turístico inteligente

Máster Universitario en Ingeniería Informática

Trabajo Fin de Máster

Autor:
David Peinado Sempere

Tutor/es:
José Norberto Mazón López

Universitat d'Alacant
Universidad de Alicante

Septiembre 2019

# Aplicación de FIWARE en un destino turístico inteligente

David Peinado Sempere

Septiembre 2019

Universidad de Alicante

# Preámbulo

Mi interés por el mundo de la electrónica viene desde pequeño, cuando con papel de aluminio, unos cables, una pila, un zumbador y, como no, las instrucciones, podía montar una alarma rudimentaria para descubrir cuando alguien entraba a mi habitación.

Unos cuantos años más tarde, ya cursando el Grado en Ingeniería Informática en la Universidad de Alicante, me inscribí en un curso de robótica con Arduino que me llamó mucho la atención. En él aprendí la gran cantidad de tipos de sensores que pueden montarse sobre esta clase de dispositivos. No obstante, no fue hasta la Campus Party de 2016, cuando asistí a una exposición de la empresa The Things Network sobre la aplicación práctica de la sensorización en un ámbito público, como pueda ser una ciudad. En ese momento la idea que daría lugar a este trabajo comenzó a formarse.

Una vez finalizada la carrera, cursando el Máster en Ingeniería Informática, hablé con José Norberto Mazón, mi tutor en este trabajo, sobre la posibilidad de hacer algo relacionado con el turismo, actividad que él trata de mejorar en la ciudad de Torrevieja (Alicante). En este momento decidimos contactar con la empresa HOP Ubiquitous, que se dedica al mundo del Internet of Things (IoT), y nos enseñaron los dispositivos que utilizan en las soluciones que implantan. Con estos conocimientos y la idea de monitorizar ciertos puntos turísticos de la ciudad empezamos a trabajar para reconocer las rutas que realizan los turistas y por qué factores se ven influenciadas.

Por tanto, en este TFM se utilizarán técnicas de diseño de software y de recopilación de datos con el fin de obtener un sistema capaz de detectar de manera anónima cuáles son los caminos que la mayoría de personas elige para visitar los diferentes puntos turísticos que se adapten en la ciudad.

Con este proyecto pretendo aprender sobre la arquitectura en los sistemas de sensorización y los estándares de la industria del internet de las cosas así como tener una primera toma de contacto con la recolección, procesamiento y análisis de datos.

## Agradecimientos

Quería dedicar este pequeño espacio para agradecer a todos aquellos que han hecho posible la realización de este trabajo.

En primer lugar, a José Norberto Mazón, no sólo por ser mi tutor en este trabajo, además, también por ser un gran profesional con el que he compartido ideas e impresiones a lo largo de mi paso por la Universidad de Alicante. Gracias a él, también llegué a conocer al personal de la empresa HOP Ubiquitous, que fueron increíblemente gentiles al explicarme cómo gestionaban sus dispositivos y datos.

No me gustaría olvidar a todo aquel que haya contribuido en la comunidad FIWARE, puesto que la implementación técnica del sistema desarrollado en este trabajo, no habría sido posible sin la ayuda de toda esa gente que dedica una fracción de su tiempo a resolver y documentar dudas y preguntas para que todos podamos solventar nuestros problemas.

Finalmente, agradecer a todos aquellos que han aportado su tiempo y dedicación de uno u otro modo para hacer posible mi formación como Ingeniero Informático.

# Índice general

# Índice de figuras

# Lista de acrónimos

**AP** punto de acceso. 10, 11

**BLE** Bluetooth Low Energy. 14

**CB** Context Broker. 18, 19, 24–27, 29, 30, 32, 35

**DTI** destino turístico inteligente. 1

**GE** generic enabler. 17, 18, 20, 23–26, 32, 40

**IoT** Internet of Things. 1, 4, 5, 17, 18, 24–26

**IOTA** IoT Agent. 25–31

**PIT** punto de interés turístico. 2, 9, 33

**PR** probe request. 10–13, 16

**SC** Smarty City. 1, 4, 5, 18, 24

**SSID** identificador de la red. 10, 11

**STD** Smart Tourism Destination. 1, 2, 18, 24, 33, 37, 40

**TIC** tecnologías de la información y la comunicación. 1

# Glosario

**Backend Device Management Generic Enabler** Elemento del ecosistema FIWARE encargado de registrar y administrar las entidades en el sistema. 24–26

**Context Consumer** Elementos que consumen la información de contexto, son aquellos que reciben las actualizaciones y realizan acciones con ellas. 20

**Context Producer** Elementos que actúan como fuente de información, son los que crean o actualizan la información de contexto. 20, 24, 25

**Context Provider** Elementos que proporcionan la información de contexto a petición del Context Broker. 20, 24

**FIWARE** Iniciativa de la Unión Europea que impulsa una plataforma de Código Abierto para el desarrollo de aplicaciones y soluciones directamente vinculadas a la Smart City. 2, 4, 5, 9, 17–19, 21, 22, 24–26, 30, 31, 39, 40

*La tecnología hizo posible las grandes poblaciones;*
*ahora las grandes poblaciones hacen que la tecnología sea indispensable*
*J. Krutch*

# 1 Introducción

La inclusión de la tecnología en el día a día de la sociedad es uno de los principales motores de crecimiento para la economía y el bienestar. Cada vez más sensores, aparatos y aplicaciones se integran en nuestra vida y nuestros alrededores para mejorarla. Las tecnologías de la información y la comunicación (TIC) han creado un ambiente favorable para el desarrollo de nuevas formas de gestionar las ciudades (Ivars-Baidal et al., 2019). Gracias a esto, se da el escenario en el que se tiene la capacidad para la incorporación de mejoras tecnológicas para estas ciudades denominadas Smart Cities (SCs) o ciudades inteligentes (Komninos, 2002). Según E. Hall et al. (2000), una SC es una ciudad que monitoriza e integra las condiciones de sus infraestructuras para optimizar sus recursos y maximizar los servicios ofrecidos a los ciudadanos.

Los aspectos a gestionar en una ciudad inteligente son diversos y, entre ellos, es de vital importancia el ámbito del turismo. En concreto, en España, los ingresos generados por el sector turístico representan un 14,6 % del producto interior bruto, según el informe anual de 2019 de World Travel & Tourism Council[1]. Esto hace que se trate de un área en la que la implantación de soluciones tecnológicas para mejorar los servicios ofrecidos pueda afectar a un gran número de personas desde el punto de vista económico y social.

Según Buhalis y Amaranggana (2015), combinar el turismo con soluciones tecnológicas inteligentes requiere conectar de forma dinámica a aquellos involucrados en la actividad a través de una plataforma en la que la información pueda ser intercambiada de forma instantánea.

Así pues, tal como dice Buhalis y Amaranggana (2013), el desarrollo de una SC podría propiciar la formacion de Smart Tourism Destinations (STDs), en castellano, destinos turísticos inteligentes (DTIs), que se aprovechan de (1) entornos tecnológicos integrados; (2) procesos sensibles a alto y bajo nivel; (3) dispositivos de usuario final en múltiples puntos de contacto; y (4) participación de las partes interesadas que utilizan la plataforma de forma dinámica como un sistema de conocimiento.

Las necesidades expuestas para la generación de STDs encajan con la idea que trata de conseguir el Internet of Things (IoT), que es generar interacciones automáticas en tiempo real con un objeto físico que se conecta a Internet con el objetivo de minimizar la distancia entre este objeto y el mundo digital (Erb, 2011). El IoT es, por tanto, una buena herramienta para la creación de las plataformas mencionadas ya que permite transmitir datos utilizando un sistema de sensorización participativo (Gutiérrez et al., 2013).

A pesar de la existencia del IoT, falta una pieza clave para poder permitir hacer uso de los datos extraídos de un STD, la comunicación con la plataforma. Existen diversas iniciativas que tratan de estandarizar y proporcionar las herramientas y conocimientos necesarios para

[1]Enlace a la página de la organización World Travel & Tourism Council (WTTC): `https://www.wttc.org/`

el desarrollo y explotación tecnológico de las urbes. Uno de estos proyectos es FIWARE[2], impulsado por la Unión Europea desde 2011, cuya misión es «construir un ecosistema abierto y sostenible [...] que facilite el desarrollo de aplicaciones inteligentes en múltiples sectores».

¿Alguna vez han ido a una ciudad y no han sabido qué lugares visitar y en qué orden? Actualmente existen diferentes soluciones a este problema, ya sea mediante el uso de aplicaciones móviles, consultando en internet o preguntando a personas locales. No obstante, cualquiera de estos métodos se basa en la opinión de un colectivo reducido que puede haber sido influenciado. Este trabajo pretende poner solución a este problema mediante la aplicación de FIWARE y así aportar datos contextuales para saber en qué situaciones se han desarrollado diversos recorridos por la ciudad.

El sistema desarrollado por este trabajo proporcionará la plataforma necesaria para considerar a un destino turístico como inteligente, en el sentido de que los datos extraídos proporcionarán la información para determinar cómo se desplazan los visitantes en un STD. Estos datos serán extraídos utilizando sensores posicionados en diferentes puntos de interés turístico (PITs) de una ciudad, de forma que se puedan identificar los diferentes itinerarios que realizan las personas que se desplazan por la ciudad.

[2]Enlace a la página del proyecto FIWARE: `https://www.fiware.org/`

# 2 Estado de la cuestión

La detección de desplazamientos en un área geográfica presenta dos principales desafíos, la identificación de dispositivos y la gestión y recolección de la información. En las siguientes secciones se habla de diversos estudios en los que se afrontan problemas similares y mediante los cuales se extraerán conclusiones para la realización de este trabajo.

## 2.1. Detección de dispositivos

Diversos estudios han abordado el problema de la detección de desplazamientos de personas en una ciudad turística. Donaire et al. (2008) propone un modelo de observación, seguimiento y realización de encuestas, no obstante, este método no es sostenible a lo largo del tiempo, dado que las tendencias turísticas se ven modificadas por múltiples factores. Law et al. (2011) ha tratado de mitigar este tipo de cambios aplicando técnicas de Data Mining a encuestas realizadas durante un periodo de cinco años para detectar las diferentes tendencias y sus modificaciones y tratar de predecir estos comportamientos entre los ciudadanos de Hong Kong, sin embargo, esta aproximación también requiere de procesos manuales para la recopilación de las encuestas y, no se garantiza que las respuestas a estas sean correctas.

Con la irrupción de las tecnologías de la comunicación y el auge por de los dispositivos móviles, se realizan estudios que tratan de realizar el seguimiento de individuos mediante el uso de las señales emitidas por sus teléfonos móviles.

Existen estudios que aplican técnicas de seguimiento de dispositivos aplicando tecnologías de lectura de paquetes en diferentes redes, este tipo de análisis se denomina seguimiento por radiofrecuencia (RF-Tracking) y se ha aplicado a tecnologías como el RFID (Rose, 2017), el Wi-Fi o el Bluetooth.

Mei et al. (2014) propone la utilización de sensores lectores de señales Bluetooth para realizar el análisis y seguimiento de trayectos en bicicleta de diferentes individuos con el objectivo de mejorar los modelos de movilidad en la ciudad de Hangzhou.

También hay estudios que tratan de utilizar las señales Wi-Fi para el seguimiento de individuos. Cunche (2014) explica, en un análisis desde el punto de vista de la seguridad y privacidad, los datos que pueden extraerse de las tramas Wi-Fi, entre otros, la MAC del dispositivo que la ha emitido. Es decir, es posible identificar inequívocamente a un dispositivo con la lectura de una trama enviada por el mismo. Musa y Eriksson (2012) y Husted y Myers (2010) realizan estudios y aplicaciones a casos reales de las técnicas de detección de individuos. El primero pone en marcha un funcionamiento mediante el que mide los costes de un sistema de detección Wi-Fi y los resultados que se obtienen con el mismo a nivel cuantitativo además de tratar de realizar una estimación de la ruta seguida por el dispositivo detectado. El segundo, hace también un análisis y una simulación en los que se determinan

las condiciones necesarias para el seguimiento de dispositivos con la lectura de tramas en entornos urbanos así como la precisión de la localización de individuos con el uso de este sistema.

En cuanto al uso del Wi-Fi y el Bluetooth, existen trabajos que analizan los tipos de paquetes que pueden ser leídos y cómo se envían los identificadores de los dispositivos en ambos protocolos (Welch y Lathrop, 2003). Además, es posible incluso establecer una correlación entre ambas señales y determinar, mediante la intensidad de la señal y los fabricantes de los chips de comunicaciones, si dos capturas en las diferentes frecuencias pertenecen al mismo dispositivo o individuo (Longo, 2017).

La vulnerabilidad de la privacidad del usuario mediante el uso de estas técnicas y los riesgos que pueden implicar han sido trabajados en diversas ocasiones en estudios como Cunche (2014), Husted y Myers (2010) o Longo (2017). De estos se extrae que la forma de evitar que un dispositivo sea identificado a lo largo del tiempo por las tramas Wi-Fi que emite es mediante el uso de la llamada randomización de MAC y, en el caso del Bluetooth, la forma mas eficaz de evitar el seguimiento es manteniendo los dispositivos en modo oculto para que así no emitan esos paquetes. Aún así, en el caso del Wi-Fi, se ha podido establecer correlación entre dos paquetes generados por un mismo dispositivo que aplica la técnica de randomización de MAC (Vanhoef et al., 2016).

## 2.2. Gestión y recolección de los datos

Los estudios anteriores tratan la temática de la captura de datos para el seguimiento de individuos, sin embargo, no se detalla la forma en la que recoger los datos y conseguir un sistema recolectando información constante.

De Poorter et al. (2011) propone un modelo de comunicación heterogéneo que permite a los dispositivos y las aplicaciones comunicarse mediante un punto de acceso común que soluciona el problema de los diferentes protocolos utilizados por los distintos miembros de una red de dispositivos IoT.

El estudio que Gubbi et al. (2013) realiza, pone en marcha un sistema de monitorización de ruido en áreas urbanas compuesto por diferentes grupos de sensores que envían datos al mismo tiempo que puede ser extendido para utilizar otro tipo de sensores.

La propuesta de Jin et al. (2014) es una arquitectura con la que desarrollar SCs proporcionando información en tiempo real al ciudadano. Este modelo tiene en cuenta los sensores y las redes que les permiten la comunicación, así como la gestión e interpretación de los datos.

Existen diferentes iniciativas y plataformas de código abierto que pretenden proporcionar los estándares necesarios para realizar las comunicaciones y gestionar los datos en un entorno sensorizado. Mehmood et al. (2017) menciona algunas de ellas y las describe del siguiente modo:

- FIWARE. Se trata de una plataforma de código abierto para el desarrollo de SCs que busca desarrollar la base de las tecnologías en el paradigma IoT. Se basa en componentes software que proporcionan funcionalidades para permitir la interoperabilidad de componentes.

- OCEAN. Es una asociación que pretende producir implementaciones de código abierto para SCs basadas en estándares IoT. Se centran en promocionar el desarrollo y comercialización de plataformas, productos y servicios que cumplan los estándares IoT.
- OM2M. Este proyecto se inició por la fundación ECLIPSE para proporcionar implementaciones de código abierto a los estándares oneM2M[1] y SmartM2M[2]. El objetivo principal del proyecto es permitir el despliegue de aplicaciones verticales y dispositivos heterogéneos proporcionando una plataforma para desarrollar servicios de forma independiente a la red en la que se encuentren.
- Open Daylight IoT data management. El objetivo principal del proyecto es el desarrollo de un componente central enfocado a manejo de datos, que representa un broker con oneM2M, y permite autorizar el acceso a los datos generados por los dispositivos.
- CONTIKI. Plataforma de código abierto que permite el desarrollo de aplicaciones para SCs basadas en IoT de forma rápida y sencilla. Ofrece comunicaciones a través de Internet para microcontroladores a costa de un coste y energía muy reducido.

## 2.3. Conclusiones para el trabajo

Como se ha podido comprobar, diferentes formas de abordar el problema de detección de desplazamientos de individuos se han utilizado en diversos estudios. Para este trabajo, se ha elegido el RF-Tracking utilizando señales Wi-Fi y Bluetooth ya que son las tecnologías más extendidas entre la mayoría de los dispositivos utilizados actualmente. Al utilizar ambos tipos de señales se pueden tener dos formas independientes de detectar dispositivos, aumentando así el número de posibles detecciones.

En cuanto a la recolección de datos de los sensores, se ha optado por el uso de FIWARE debido a la cantidad de documentación disponible, las facilidades aportadas para desarrolladores en forma de componentes software que pueden comunicarse entre sí y como motivación personal para conocer uno de los estándares más nombrados a nivel europeo en el ámbito de las SCs.

[1] Enlace a las especificaciones oneM2M: `http://www.onem2m.org/`

[2] Enlace a las especificaciones SmartM2m: `https://portal.etsi.org/TBSiteMap/SmartM2M/SmartM2MToR.aspx`

# 3 Objetivos

La realización del trabajo establece como objetivo general:

- O.G.1: Desarrollar un sistema para la detección de desplazamiento de individuos aplicando FIWARE.

Además, será necesario plantear diferentes objetivos específicos que ayudarán a alcanzar el objetivo general:

- O.E.1: Desarrollar el sistema haciendo uso de herramientas de código abierto..
- O.E.2: Desarrollar el sistema de forma que se garantice la anonimidad de los datos.
- O.E.3: Desarrollar el sistema de forma que los datos puedan ser publicados en plataformas de datos abiertos.
- O.E.4: Desarrollar el sistema de forma que la toma de datos esté separada del procesamiento y análisis de los mismos.
- O.E.5: Desarrollar el sistema de forma que pueda ser ampliado fácilmente.
- O.E.6: Desarrollar el sistema de forma que pueda ser puesto en funcionamiento de forma sencilla.

# 4 Metodología y tecnologías

La realización de este trabajo pasa por diferentes etapas a través de las cuales se intenta cumplir los objetivos específicos para así alcanzar el objetivo principal establecido.

Las etapas se pueden dividir en las siguientes:

1. Fase de análisis y planificación: se analizan las tecnologías con las que se va a trabajar y se planifica el modo de realizar la implementación.
2. Fase de implementación: se sigue la planificación establecida en la fase anterior para lograr implementar el código necesario para la puesta en funcionamiento del sistema.
3. Fase de puesta en funcionamiento: se realizan las tareas necesarias para poner en funcionamiento el sistema implementado.
4. Fase de comprobación de resultados: con el sistema ya funcionando, se empiezan a recopilar los primeros datos.

En la Sección 4.1: Wi-Fi y la Sección 4.2: Bluetooth, se explica con detalle el análisis realizado para cada una de las tecnologías. Además, en estas secciones se especifica la implementación que se ha realizado y las herramientas utilizadas para extraer la información necesaria de los dispositivos cercanos a los puntos sensorizados.

Posteriormente, en la Sección 4.3: FIWARE, se ha hecho un análisis de la iniciativa FIWARE para conocer sus elementos a nivel de arquitectura y componentes. Con esta información ha sido posible averiguar qué tipo de datos se pueden almacenar, dónde y cómo. Se han elegido los componentes que se utilizarán y así se ha conseguido establecer la forma en la que se capturan los datos en los PITs sensorizados.

En el Capítulo 5: Sistema de detección de desplazamientos, se expone el sistema final en funcionamiento, a nivel de arquitectura y a nivel de flujos de datos. Seguidamente, se muestran posibles usos de los datos extraídos para realizar representaciones gráficas de los desplazamientos.

Finalmente, en el Capítulo 6: Conclusión, se comprueban los objetivos iniciales establecidos para determinar si se han podido cumplir y se marcan las limitaciones y líneas futuras y posibles extensiones para el sistema.

Tras establecer las pautas con las que se ha estructurado el trabajo, se comienza con el análisis de las diferentes tecnologías utilizadas para la construcción del sistema propuesto.

## 4.1. Wi-Fi

En esta sección se exponen los resultados de la investigación sobre la tecnología Wi-Fi que era necesaria con el fin de realizar la posterior implementación de los programas usados por

el sistema para detectar dispositivos.

### 4.1.1. IEEE 802.11

Las comunicaciones Wi-Fi son definidas en el estándar IEE 802.11 (Committee et al., 1999). Las especificaciones menos recientes (802.11b/g) operan en la frecuencia de 2.4GHz sobre 11 canales de los cuales solo tres no se solapan, el 1, el 6 y el 11 (ver figura 4.1[1]). En la especificación más reciente, la 802.11n, se establecen nuevos canales y una nueva frecuencia de operación, la de 5GHz.

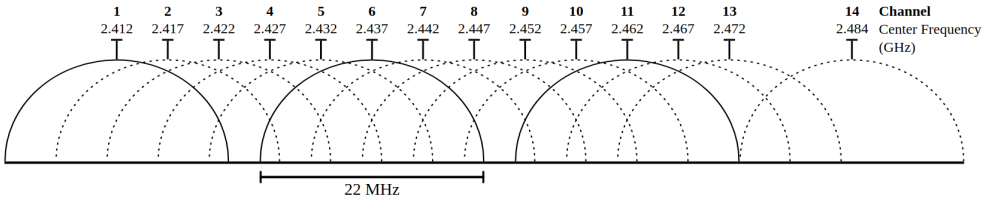


Figura 4.1: Canales Wi-Fi 2.4GHz (802.11b,g).

La mayoría de dispositivos modernos cambian entre 802.11/b/g/n dependiendo de las demandas de tráfico, la congestión de la red y la distancia al punto de acceso (AP) Freudiger (2015).

La organización IEEE define en el estándar Wi-Fi 802.11 (Committee et al., 1999) dos formas de descubrir dispositivos y APs. Los APs indican su presencia mediante los Beacons (ver figura 4.2a), y los dispositivos utilizan las probe requests (PRs) para localizar a los APs (ver figura 4.2b).


(a) Descubrimiento pasivo de APs Wi-Fi. (b) Descubrimiento activo de APs Wi-Fi.

Figura 4.2: Formas de detectar dispositivos con señales Wi-Fi.

#### Beacons

Los APs a Wi-Fi anuncian su presencia mediante el envío de paquetes, llamados Beacons, que contienen parámetros de configuración de la red así como el identificador de la red (SSID) e información acerca de las limitaciones de envío de datos. Los dispositivos móviles están a la escucha de Beacons en los canales 802.11b/g/n y detectan de forma pasiva los APs cercanos. La frecuencia con la que estos paquetes son enviados depende de la configuración

[1] Autor Michael Gauthier, Wireless Networking in the Developing World

de descubrimiento de la red y del ancho de banda. Muchos APs establecen un intervalo de 100ms para el envío de beacons y los dispositivos responden a estos paquetes con paquetes de asociación (Freudiger, 2015).

### Probe requests

Los dispositivos pueden buscar APs de forma proactiva mediante el envío de PRs. Estos paquetes incluyen, entre otros, un identificador único del dispositivo y las operaciones que soporta así como el SSID de la red que busca en caso de que esté buscando una en concreto. No todos los PRs llegan a los APs, pero en caso de llegar, el AP responderá con una probe response. (Freudiger, 2015).

El uso práctico de estos paquetes reside en que los dispositivos pueden encender la radio Wi-Fi por un breve periodo de tiempo, enviar los PRs y determinar si hay redes próximas a las que poder conectarse. De este modo se ahorra energía, se puede mantener la conexión activa mientras el dispositivo está durmiendo y, además, es la única forma de conectarse a redes ocultas.

Las PRs, por tanto, permiten identificar dispositivos puesto que contienen la dirección MAC y, en ocasiones, más información que será omitida por el sistema puesto que el objetivo es únicamente detectar dispositivos, no la identificación de personas o datos sensibles.

### Estructura de paquete

Los paquetes que se pretenden analizar para extrar el identificador del dispositivo constan, según Stallings (2005) de diferentes bloques (ver figura 4.3):

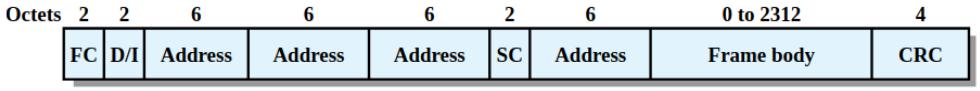


Figura 4.3: Formato paquete MAC 802.11 (Autor William Stallings "Data and Computer Communications").

- FC. Indica el tipo de paquete y información de control.
- Address. Dirección del emisor, transmisor y destinatario.
- SC. Identificador del paquete entre un emisor y un receptor.
- Frame body. Contiene la información a transmitir.
- CRC. Código de redundancia cíclica.

Entre estos bloques, tan solo es necesario prestar atención al FC y al Address 1, que indicarán el tipo de paquete que se quiere analizar, en este caso, las PRs y la dirección MAC del dispositivo emisor. Extrayendo la mínima información del paquete se pretende evitar vulnerar la privacidad del usuario.

### 4.1.2. Detección de dispositivos con Wi-Fi

En esta sección se expone la implementación realizada para detectar dispositivos enviando PRs así como las librerías utilizadas para facilitar este trabajo.

#### Librerías utilizadas

La implementación del script de detección de dispositivos se ha realizado mediante la ayuda de paquetes con licencias abiertas para el uso.

- Scapy[2]. Licenciado bajo GNU General Public. Se trata de una librería para el manejo y manipulación de paquetes, con ella ha sido posible utilizar una interfaz de red en modo monitor para analizar el tráfico de PRs cercanas.
- Beaker[3]. Bajo licencia BSD License (BSD). Este paquete permite utilizar una caché en disco, esto ha sido utilizado para determinar un tiempo mínimo de registro entre la detección de diferentes PRs del mismo dispositivo. Así se evita capturar varias tramas de golpe.
- Pytz[4]. Bajo licencia MIT License (MIT). El uso de este paquete ha sido para poder añadir a la información capturada la fecha y la hora con información de zona horaria UTC.

#### Implementación

El programa implementado para capturar tramas Wi-Fi está hecho con lenguaje Python en su versión 3.7.4.

El funcionamiento del programa es el siguiente:

1. Iniciar escucha de paquetes Wi-Fi.
2. Detección de trama, por cada trama hacer:
   *a*) Obtener fecha y hora de la detección.
   *b*) Extraer MAC del origen del paquete.
   *c*) Calcular el hash con el algoritmo SHA256 de la MAC.
   *d*) Comprobar si el hash está en la caché de dispositivos recientes.
   *e*) En caso de estar, se detiene el procesamiento, si no está, se añade a la caché y se registran los datos leídos.

La caché de dispositivos recientes se utiliza para evitar que un mismo dispositivo sea detectado diversas veces en cortos periodos de tiempo. De este modo, una vez se realiza una detección, se almacena el identificador de la misma y hasta que no transcurra el tiempo

[2]Enlace a Pypi del paquete Scapy: `https://pypi.org/project/scapy/`
[3]Enlace a Pypi del paquete Beaker: `https://pypi.org/project/Beaker/`
[4]Enlace a Pypi del paquete Pytz: `https://pypi.org/project/pytz/`

configurado para la persistencia de la caché, ese mismo dispositivo no volverá a ser registrado por el programa.

En cuanto al uso de un algoritmo para calcular el hash de la MAC del dispositivo, esto se realiza por protección de privacidad. Este tipo de algoritmos son unidireccionales, es decir, a partir de un dato de entrada, siempre se obtiene el mismo dato de salida, no obstante, actualmente, a partir del dato de salida es computacionalmente poco probable obtener el dato original. Registrando sólo el resultado de esta función, evitamos la identificación de un aparato y, de este modo, se evita que un usuario pueda ser reconocido.

Los datos que extrae por cada trama capturada contienen el identificador del dispositivo, el tipo de trama que se captura, en este caso, siempre el valor `wifi`, y el tiempo en el que se capturó (ver bloque de código 4.1).

```
{
   "source":"wifi",
   "device_identifier":"
      a7dee577d9aa4a7608d56b1e46e8bd76ea803f8bf20eb9eb6f5f61685c7fef8f
      ",
   "time":"2019-08-23T19:29:08.928991+00:00"
}
```

Código 4.1: Datos extraídos por el script de detección de dispositivos de una PR.

Cabe destacar que el procesamiento de las tramas se realiza de forma individual y asíncrona. Esto permite que el proceso principal pueda seguir capturando PRs para no perder posibles registros y, cada vez que se detecta una trama, se invoca un subproceso que se encarga de registrar el identificador del dispositivo.

## 4.2. Bluetooth

En esta sección se exponen los resultados de la investigación sobre la tecnología Bluetooth, necesaria para la posterior implementación de los scripts usados por el sistema para detectar dispositivos.

### 4.2.1. IEEE 802.15

Las comunicaciones Bluetooth son definidas en el estándar IEEE 802.15. Esta tecnología transmite ondas de radio de baja energía en la frecuencia de 2.4GHz realizando saltos entre frecuencias para evitar interferencias (ver figura 4.4). Este espacio de frecuencias está reservado internacionalmente para el uso industrial, científico y médico (ISM) y no requiere licencias para dispositivos que lo utilicen.

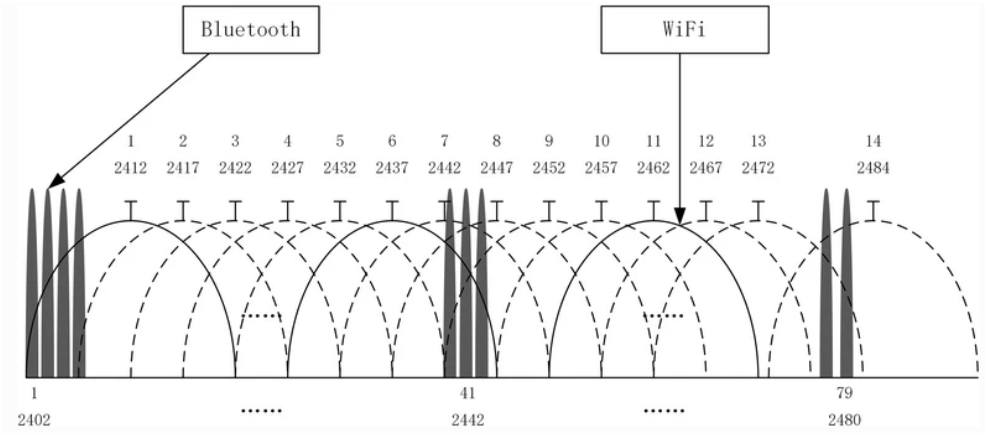


Figura 4.4: Espectro de frecuencias combinado de Wi-Fi Bluetoothh (Pei et al., 2017).

Existen diferentes revisiones de las especificaciones Bluetooth, y no es hasta la versión 4.0 que aparece la especificación del Bluetooth Low Energy (BLE), estándar que está optimizado para dispositivos pequeños con restricciones de consumo energético. En la especificación de Bluetooth 5, se incorporan mejoras de rendimiento que aumentan el rango de comunicación hasta los 780 metros (Becker et al., 2019).

El BLE opera en 40 canales desde la frecuencia 2402MHz hasta 2480MHz, pero sólo 3 de estos, 2402MHz, 2426MHz y 2480MHz se usan para anunciar (Heydon, 2013). Estos canales tienen como propósito servir para anunciar mensajes de diferentes tipos como anuncios de presencia, peticiones de escaneo a otros dispositivos, beacons con información de localización y otros tipos de información pública. En la capa de enlace de Bluetooth se definen 5 estados y tan solo en el estado de anunciación se generan mensajes, que se transmiten por todos estos canales, y son escuchados por los dispositivos en modo escaneo (Becker et al., 2019).

Las mayores diferencias entre Bluetooth clásico y BLE son el consumo energético, el alcance y que este último pasa la mayor parte del tiempo en modo sueño y sólo despierta cuando se inicia una conexión.

Los mensajes de BLE pueden ser mensajes de anunciación o mensajes de datos, como se menciona anteriormente, cada uno se envía a través de canales diferentes y también presentan estructuras diferentes (ver figura 4.5).

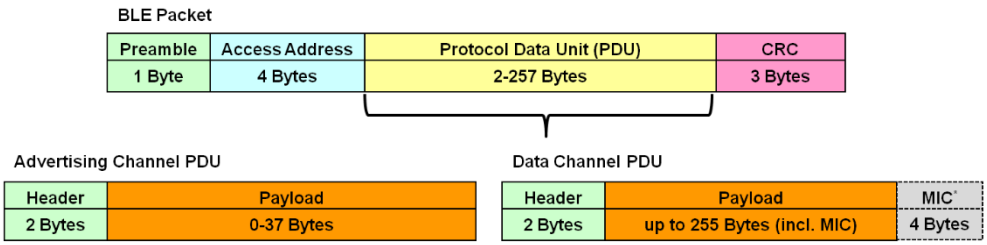


Figura 4.5: Tipos de mensajes BLE (fuente Microchip Developer)[5].

Los mensajes que anuncian dispositivos, a su vez, pueden ser de diferentes tipos, de anuncio, de escaneo o de inicio de conexión. Esta información esta presente en la cabecera (los dos primeros bytes) de estos mensajes. El resto de la información está en el bloque de datos, del cual se puede extraer la dirección del dispositivo que emite el mensaje. Jameel y Dungen

(2015) expone diversos modos de extraer la información de los mensajes BLE.

En cuanto al formato de los paquetes de Bluetooth clásico, cada uno se divide en tres bloques, el código de acceso, la cabecera y los datos. El primero, access code en inglés, se utiliza para sincronización, ajuste de desplazamiento e identificación (ver figura 4.6).

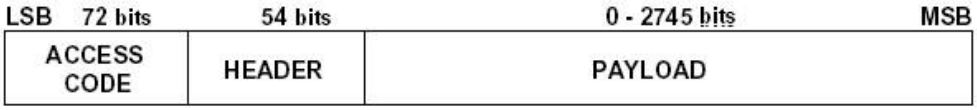


Figura 4.6: Formato del bloque ACCESS CODE de un paquete bluetooth (SIG, 2001).

Este primer bloque de los mensajes está a su vez dividido en tres bloques, preamble, sync word y trailer. El bloque interesante para el sistema que se pretende desarrollar es el segundo, el sync word, puesto que es un código de 64 bits derivado del LAP. El LAP es una parte de la dirección Bluetooth determinada por el fabricante que permite identificar a un dispositivo (ver figura 4.7).

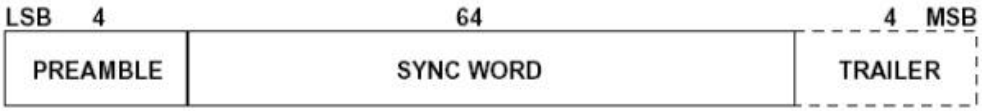


Figura 4.7: Formato del bloque ACCESS CODE de un paquete bluetooth (SIG, 2001).

Por tanto, mediante la lectura de los bloques analizados, tanto para Bluetooth clásico como para BLE, será posible la identificación de los dispositivos cercanos al sensor. Cabe destacar que ambas tecnologías, son relevantes para el sistema puesto que se tratará de detectar dispositivos con cualquiera de ellas y, de este modo, podrán identificarse una mayor cantidad de individuos.

### 4.2.2. Detección de dispositivos con Bluetooth

En esta sección se expone la implementación realizada para detectar realizando peticiones Bluetooth así como las librerías utilizadas para facilitar este trabajo.

#### Librerías utilizadas

La implementación del script de detección de dispositivos se ha realizado mediante la ayuda de paquetes con licencias abiertas para el uso.

- Pybluez[6]. Licenciado bajo GNU General Public. Se trata de una librería que permite hacer uso de los recursos Bluetooth del sistema a través de Python, el lenguaje elegido para implementar la detección de dispositivos. Gracias a esta librería, no ha sido necesario trabajar con los paquetes a nivel de bit, puesto que propociona funcionalidades para obtener directamente la dirección de un dispositivo.
- Beaker[7]. Bajo licencia BSD License (BSD). Este paquete permite utilizar una caché en disco, esto ha sido utilizado para determinar un tiempo mínimo de registro entre la

[6]Enlace a Pypi del paquete Pybluez: `https://pypi.org/project/PyBluez/`
[7]Enlace a Pypi del paquete Beaker: `https://pypi.org/project/Beaker/`

detección de diferentes PRs del mismo dispositivo. Así se evita capturar varias tramas de golpe.

- Pytz[8]. Bajo licencia MIT License (MIT). El uso de este paquete ha sido para poder añadir a la información capturada la fecha y la hora con información de zona horaria UTC.

#### Implementación

El programa implementado para capturar los mensajes Bluetooth está hecho con lenguaje Python en su versión 2.7.15 debido a una incompatibilidad de una de las librerías.

El funcionamiento del programa es el siguiente:

1. Iniciar escucha de paquetes Bluetooth clásico.
2. Detección de mensaje, por cada mensaje hacer:
   *a*) Obtener fecha y hora de la detección.
   *b*) Extraer MAC del origen del paquete.
   *c*) Calcular el hash con el algoritmo SHA256 de la MAC.
   *d*) Comprobar si el hash está en la caché de dispositivos recientes.
   *e*) En caso de estar, se detiene el procesamiento, si no está, se añade a la caché y se registran los datos leídos.

Tal como se explicaba para el programa que detecta dispositivos Wi-Fi, en este, la caché de dispositivos recientes también se utiliza para evitar que un mismo dispositivo sea detectado diversas veces en cortos periodos de tiempo.

De igual modo, la forma de anonimizar la identidad del dispositivo detectado es haciendo uso del algoritmo SHA256 sobre la MAC detectada.

Los datos que extrae por cada trama capturada contienen el identificador del dispositivo, el tipo de trama que se captura, en este caso, siempre el valor `wifi`, y el tiempo en el que se capturó (ver bloque de código 4.2).

```
{
   "source":"bluetooth",
   "device_identifier":"
      d1d4170ae34154f53aab0625225def6dd9613b84f1d0479ad94c3dd68349cd1c
      ",
   "time":"2019-08-24T16:03:14.922289+00:00"
}
```

Código 4.2: Datos extraídos por el script de detección de dispositivos de un paquete Bluetooth.

[8]Enlace a Pypi del paquete Pytz: `https://pypi.org/project/pytz/`

En este caso, existe una diferencia con el programa Wi-Fi, para poder detectar dispositivos utilizando Bluetooth clásico y BLE, este programa lanza dos procesos, cada uno de ellos escaneando mensajes de las diferentes tecnologías, para así evitar tiempos de espera en los que no se está escaneando alguno de los dos tipos de mensajes.

## 4.3. FIWARE

El proyecto FIWARE, tal como se define en su página web [9], es una iniciativa de origen abierto que establece una serie de estándares para el manejo de datos de contexto.

El contexto en el ámbito de la informática es definido como cualquier información que puede usarse para determinar la situación de una entidad. Como entidad se entiende una persona, un lugar o un objeto relevante para la interacción entre un usuario y una aplicación incluyendo al usuario y a la aplicación (Dey, 2001).

Las soluciones que incorporan FIWARE tienen la necesidad de recolectar, administrar y procesar información de contexto para poder actuar en consecuencia a ella. Por ello se define el flujo de información en forma de Capturar-Procesar-Actuar (ver figura 4.8).

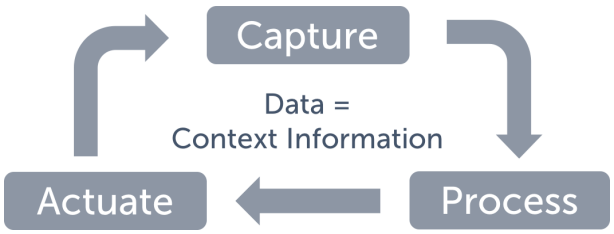


Figura 4.8: Flujo de información descrito por FIWARE. (Fox, 2019).

La iniciativa proporciona una serie de componentes llamados generic enablers (GEs) que integran interfaces para la comunicación con desarrolladores (APIs) y para la comunicación con otros GEs. Estos componentes son reutilizables, ofrecen funciones comunes a múltiples casos de uso y se dividen en las siguientes categorías (Javierlucio, 2016):

- Alojamiento en la nube. Proporcionan una infraestructura basada en cloud que permite desarrollar, desplegar y administrar servicios.
- Administración de datos y contexto. Permiten personalizar las aplicaciones y servicios de modo que utilizando diversos GEs se da la capacidad de reunir, publicar, intercambiar, procesar y analizar grandes cantidades de datos de forma rápida y eficiente.
- IoT. Hace que cualquier objeto físico ligado a un objeto digital, cuyos parámetros digitalizados estén enlazados con los valores de sensores, esté disponible, sea accesible, sea posible su búsqueda y se puedan consultar sus datos.
- Aplicaciones, servicios y distribución de datos. Dotan la capacidad de crear un ecosistema de aplicaciones, servicios y datos de forma que se crea el escenario idóneo para la construcción de nuevas formas de explotar la información.

[9]Enlace a la página web de desarrolladores del proyecto FIWARE: `https://www.fiware.org/developers/`

- Interfaces para redes y dispositivos (I2ND). Permiten utilizar una infraestructura de red estandarizada y abierta.
- Interfaces web de usuario avanzadas. Proporcionan las herramientas necesarias para dar visibilidad a los usuarios finales y mejorar la experiencia de usuario.
- Seguridad. Permiten realizar controles de identidad y acceso de forma que el entorno es menos vulnerable a posibles ataques maliciosos.

Además de los GE, también se definen los Specific Enablers, unos componentes similares que ofrecen funciones relevantes para dominios específicos, por ejemplo, para agricultura, para fábricas o para SCs.

El componente principal, con cuyo uso se considera que una solución utiliza FIWARE, está dentro de la categoría de administración de datos y contexto, se trata del FIWARE Context Broker (CB). Por tanto, este es uno de los GEs que va a ser utilizado para el desarrollo del sistema de detección de desplazamientos. De las revisiones existentes de las especificaciones, se ha elegido la versión 4 por ser la más actualizada y la que más componentes incluye.

Además, se necesita poder capturar los datos que identifiquen de forma anónima a los individuos que pasen cerca de los STD, por lo que se hará uso de los GEs de la categoría IoT de modo que sea posible la recolección de esta información.

En las siguientes secciones, se analizan los GEs de las categorías Administración de datos y contexto e IoT para determinar cuáles de ellos se emplearán en la solución final. Por último, se añade una sección en la que se explica la forma de integrar los detectores de dispositivos desarrollados en el ecosistema FIWARE.

### 4.3.1. Administración de datos y contexto

Los GEs para la administración de datos y contexto proporcionan diversas funcionalidades que permiten aprovechar al máximo los datos de contexto. Según Joostbr (2016) estos GEs dan la capacidad de:

- Generar suscripciones a las actualizaciones de datos y preguntar por información de contexto proveniente de diferentes fuentes.
- Modelar los cambios en el contexto como eventos que pueden procesarse para detectar situaciones que puedan requerir acciones o la generación de más información de contexto.
- Procesar grandes cantidades de datos agregando valores para generar más conocimiento y para interactuar con las entidades del sistema.
- Procesar torrentes de datos provenientes de diferentes fuentes para su posterior análisis y explotación.
- Administrar información de contexto de forma estándar. Por ejemplo, los datos de localización de las entidades.

- Administrar y publicar open data en tiempo real.
- Utilizar datos y recursos existentes para proporcionar más información a otras aplicaciones.

La información de contexto puede tener su origen en sistemas ya existentes, interacciones de usuarios o valores leídos por sensores, es por esto que surge la necesidad de una solución que permita integrar los servicios actuales y futuros de manera no intrusiva.

Este papel es cubierto por el CB, el elemento indispensable para una solución FIWARE. Su principal propósito es el de desacoplar el envío del consumo de datos a diferentes niveles. De este modo, los dispositivos le mandan actualizaciones de contexto y las aplicaciones los consumen de forma activa, mediante peticiones a la API, o de forma pasiva, mediante suscripciones que guardan el enlace al que mandar los datos (ver figura 4.9). Considerando a los dispositivos como la capa más baja y a las aplicaciones como la más alta, se establecen dos interfaces en el CB, una para consumo de datos y realizar acciones, que se denomina Northbound Interface y otra para la actualización de datos de contexto, denominada Southbound Interface.

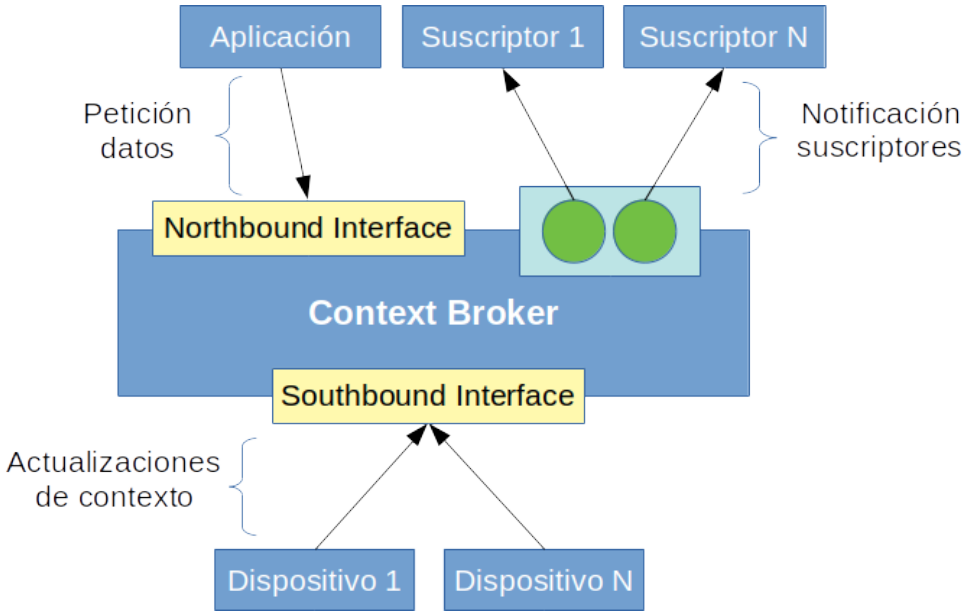


Figura 4.9: Esquema del funcionamiento del context broker.

El CB es capaz de gestionar las actualizaciones de datos de múltiples entidades, sin embargo, no almacena un histórico de esto, tan solo se encarga de actualizar los valores sustituyendo a los anteriores que tenía la entidad. Gracias al sistema de notificaciones que implementa, es posible enlazar las actualizaciones de contexto con otros servicios que se encargan de persistir estos cambios de contexto reflejados en los atributos de las entidades.

En las siguientes secciones se proporcionan detalles técnicos del funcionamiento e implementación del CB recomendado por FIWARE, el Orion Context Broker[10] y se determinan cómo son los datos que van a ser enviados y de qué forma almacenar el histórico de las

[10]Enlace al repositorio del proyecto Orion: `https://github.com/telefonicaid/fiware-orion`

actualizaciones de contexto para su posterior consumo y explotación a través de Cygnus[11].

### Orion Context Broker

Este GE permite administrar el ciclo de vida completo de la información de contexto, actualizaciones, consultas, registros y suscripciones. Implementa una API con la especificación NGSI, actualmente está disponible la version NGSIv1, la NGSIv2 y, en fase beta, NGSI-LD. Para este proyecto, se ha elegido utilizar la versión NGSIv2[12] puesto que es la más reciente que tiene versión estable.

El estándar NGSI define una API de tipo Restful con la cual se pueden realizar acciones sobre entidades y atributos. Además utiliza el protocolo HTTP permitiendo el envío de datos en formato JSON. El elemento principal en esta especificación son las entidades y sus atributos y para poder crearlos, actualizarlos y consultarlos, se definen diferentes endpoints de la API que dan esta funcionalidad. Según las acciones que realizan, aquellos sistemas o dispositivos que crean o actualizan entidades y/o atributos, se denominan Context Producers, aquellos que los consulten, Context Consumers y aquellos que pueden realizar acciones Context Providers.

Los Context Consumers pueden consumir información de contexto de dos formas, mediante una consulta convencional a la API o a través de una suscripción, previamente configurada, que manda notificaciones cuando los atributos registrados se ven modificados. Los Context Producers introducen, modifican o borran información en el sistema mediante peticiones HTTP POST/PUT/PATCH/DELETE.

Las notificaciones son completamente configurables, lo que hace posible que cada sistema participante se suscriba únicamente a los eventos que le son relevantes. Por ejemplo, se pueden configurar las alertas para que sean enviadas cuando un sensor de presión vea su valor modificado, pero tan solo se quiere recibir el atributo temperatura (ver bloque de código 4.3.

```
{
  "description": "Suscripcion para obtener valores del sensor S1",
  "subject": {
    "entities": [
      {
        "id": "S1",
        "type": "Sensor"
      }
    ],
    "condition": {
      "attrs": [
        "pressure"
      ]
    }
  },
  "notification": {
    "http": {
      "url": "http://direccion-donde-enviar-datos"
```

[11]Enlace al repositorio del proyecto Cygnus: `https://github.com/telefonicaid/fiware-cygnus`
[12]Especificación API NGSI v2: `https://fiware.github.io/specifications/ngsiv2/stable/`

```
19      },
20      "attrs": [
21        "temperature"
22      ]
23    },
24    "expires": "2040-01-01T14:00:00.00Z"
25  }
```

Código 4.3: Ejemplo suscripción con condiciones y atributos seleccionados con NGSIv2.

Existen diversas limitaciones con las acciones que pueden realizarse con HTTP y URLs Restful. Una de ellas es la limitación para realizar actualizaciones de múltiples entidades y la otra es la de realizar consultas complejas que permitan aplicar filtros.

Para solucionar la primera limitacion, NGSI incorpora una serie de URLs que permiten ejecutar actualizaciones o consultas de varias entidades a la vez, a esto se le llama batch operation.

En el caso de la segunda limitación, la solución es más compleja pero a la vez, más potente; se ha establecido un lenguaje de consultas llamado Simple Query Language que permite recuperar entidades que cumplan ciertas condiciones. Además, como la información geográfica tiene gran importancia en un sistema especialmente diseñado para obtener datos de contexto, también es posible realizar consultas mediante parámetros geoespaciales, ya sea por distancia a un punto, por localización exacta o por posicionamiento en una zona concreta.

#### Modelado de datos

El modelado de datos es uno de los puntos clave en el ecosistema FIWARE puesto que establece y promueve unas directrices a seguir para definir las entidades del mundo real junto con sus atributos de forma que sean susceptibles a ser reutilizados.

La idea principal es utilizar un modelado lo más sencillo posible, con esto en mente, se dan ciertos consejos a tener en cuenta (Federico M. Facca, 2018):

- Utilizar sólo las partes del modelo de datos que se necesitan para reducir el intercambio de información.
- Evitar el uso de metadatos siempre que sea posible, se prefiere el uso de magnitudes estándar y así no se emplear medidas personalizadas.
- Reutilizar los modelos de datos ya existentes.

Las entidades según FIWARE están definidas como elementos de contexto que tienen relación con atributos que, a su vez, están relacionados con meta datos que aportan información sobre los tipos de cada uno de los atributos (ver figura 4.10).

El modelo mínimo que puede construirse contiene un identificador, un tipo y un atributo. Este modelo puede representarse con la forma extendida (ver bloque de código 4.4) o con la forma clave valor (ver bloque de código 4.5). La última es preferible en caso de no tener atributos con datos no estándar ya que permite enviar peticiones con un cuerpo de menor tamaño.

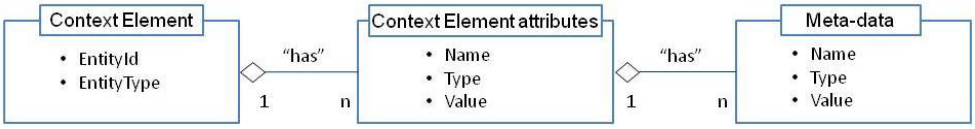


Figura 4.10: Modelo de los elementos de contexto (Joostbr, 2016).

```
{
    "id": "entityId",
    "type": "entityType",
    "att1": {
        "value": "value1",
        "type": "Text",
        "metadata": {
            "metada1": {
                "value": "metavalue1"
            }
        }
    }
}
```

Código 4.4: Ejemplo entidad en formato extendido en NGSIv2.

```
{
    "id": "entityId",
    "type": "entityType",
    "att1": "value1"
}
```

Código 4.5: Ejemplo entidad en formato clave valor en NGSIv2.

Existen una serie de atributos que son obligatorios para todos los modelos de datos según las especificaciones NGSIv2, estos son:

- El identificador único de la entidad (id).
- El tipo de que define a la entidad (type).
- La última fecha de modificación (dateModified).
- La fecha de creación de la entidad (dateCreated).

Tal como se cita anteriormente, FIWARE intenta homogeneizar el modelado de datos y, al intentar ser una solución global para integrar datos de dispositivos inteligentes o sensores, es muy probable que estos estén localizados en un punto geográfico que, en ocasiones, es de interés para la lectura e interpretación de los datos. Es por esto, que se definen tipos de datos estándar para la información de geolocalización. Las entidades pueden tener atributos que determinan la dirección postal (address) y la localización geográfica (location) en formatos estándar (ver bloque de código 4.6).

```
{
    "id": "room-001",
    "type": "Room",
    "dateCreated": "2019-08-28T19:22:51Z",
    "dateModified": "2019-08-28T19:22:51Z",
    "location": {
        "type": "Polygon",
        "coordinates": [[[25, 0], [26, 0], [26, 1], [25, 1], [25, 0]]]
    },
    "address": {
        "addressLocality": "Elche",
        "postalCode": "03400",
        "streetAddress": "Calle falsa 123"
    }
}
```

Código 4.6: Ejemplo entidad con atributos estándar de localización.

### Cygnus

El GE elegido para almacenar la información de contexto generada por los dispositivos en bases de datos y sistemas de persistencia permitiendo así la generación de un histórico de contexto es Cygnus. Este software se basa en Apache Flume, un sistema de flujo de datos que permite manejar grandes cantidades de información, transformarlas y comunicarse con otros servicios para enviar el resultado.

La arquitectura que sigue este GE consta de tres partes, una primera llamada source, encargada de recibir los datos, una segunda llamada channel, cuyo propósito es almacenar los eventos generados a partir de los datos recibidos y, una tercera y última parte llamada sink, que se encarga de procesar eventos y guardar los datos en un almacenamiento externo (ver figura 4.11).

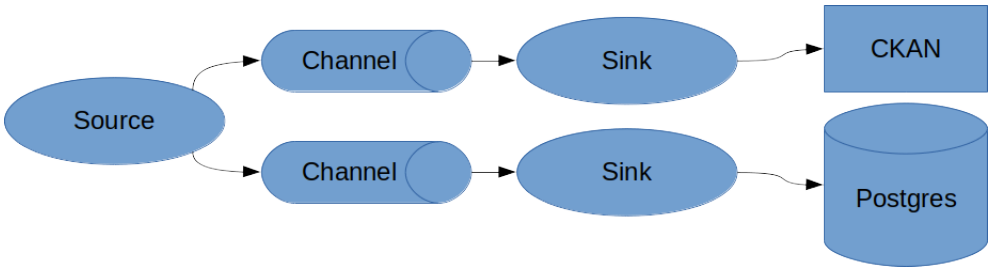


Figura 4.11: Componentes del flujo de datos de Apache Flume.

Las posibilidades que este GE proporciona permiten, por ejemplo, guardar los datos en una base de datos relacional para posteriores consultas o, incluso, generar conjuntos de datos y publicarlos en tiempo real. Entre los diferentes adaptadores para el almacenamiento en diferentes medios, se ha decidido utilizar el adaptador de PostgreSQL y el de CKAN por su fácil configuración y por la posibilidad de generar un histórico de datos que posteriormente

pueda ser utilizado, por ejemplo, para el cálculo de diferentes rutas entre los diferentes STDs de de una SC.

### 4.3.2. IoT

En la categoría IoT de GEs se proporcionan los medios para que las aplicaciones basadas en FIWARE interactúen con objetos de la vida real. Estos objetos, normalmente tienen diferentes parámetros que están total o parcialmente ligados a sensores y actuadores.

Uno de los puntos disruptivos de la idea de FIWARE es que se permite el acceso a los objetos a través de una API a los desarrolladores de aplicaciones (Gprivat, 2018). En esta categoría, por tanto, se plantean las diferentes herramientas necesarias para evitar tener que preocuparse por la forma en la que comunicarse con los objetos.

Dentro de los GEs de IoT se diferencian dos niveles el IoT Backend y el IoT Edge (ver figura 4.12. En las siguientes secciones se detalla la funcionalidad de cada uno de estos niveles de GEs.

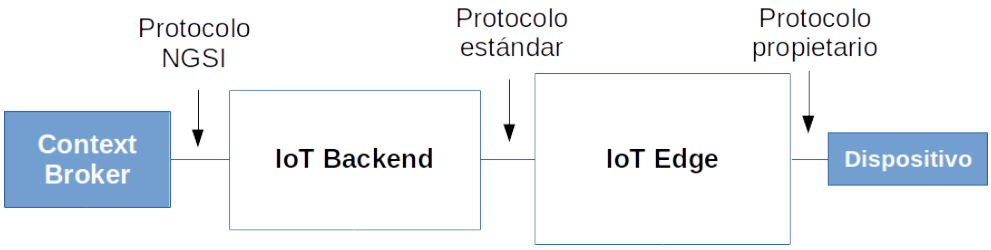


Figura 4.12: Diagrama de comunicación de los GEs de IoT.

#### IoT Backend

Exponen las funciones, los recursos y los servicios actuando de puente entre la capa física de la infraestructura IoT y la capa de datos con el CB.

Tal como se expone en Gprivat (2018), las funcionalidades del nivel de IoT Backend GEs son:

- Aprovisionamiento de las entidades. Crea una entidad NGSI por cada uno de los recursos IoT disponibles. Esta acción suele realizarse por el Backend Device Management Generic Enabler pero puede ser realizada por otros componentes en escenarios complejos. El encargado de crear las entidades toma el rol de Context Producer para todos los atributos de las entidades relacionadas a las mediciones de los recursos IoT. Además, también se autoregistra como Context Provider para aquellas entidades con funciones de actuación para, posteriormente, recibir las peticiones del CB y ejecutar el comando en el dispositivo.

- Adaptación de protocolo IoT. Hoy en día existe una amplia variedad de protocolos de comunicación para los dispositivos IoT. FIWARE proporciona los estándares de comunicación y facilita la adaptación de protocolos propietarios. La función de traducción de protocolos se realiza por el Backend Device Management Generic Enabler. Existen

herramientas que permiten realizar la traducción de algunos protocolos para algunas plataformas de hardware.

- Administración de IoT Edge. El Backend Device Management Generic Enabler también es capaz de administrar las funciones de configuración, operación y monitorización de los dispositivos ofreciendo una API adicional. De este modo, se facilita a los administradores de dispositivos poder gestionar estas acciones utilizando esta interfaz común.

- Composición y descubrimiento de dispositivos IoT. Existe la posibilidad de combinar la información de diversos sensores para generar información de contexto, además, también es necesario tener funciones de descubrimiento para encontrar nuevos dispositivos. Estas funcionalidades las proporciona el IoT Discovery GE.

La configuración mínima de un Backend Device Management Generic Enabler en el ecosistema FIWARE incluye al menos un IoT Agent (IOTA). Esto es un módulo de software que traduce entre protocolos específicos de IoT y NGSI.

Las funciones que proporciona un IOTA son las de manejo de creación de entidades NGSI en el CB por cada uno de los dispositivos conectados, realizar la función de Context Producer para los atributos relacionados con lecturas de sensores u observaciones y proporcionar una API de administración y configuración (ver figura 4.13).

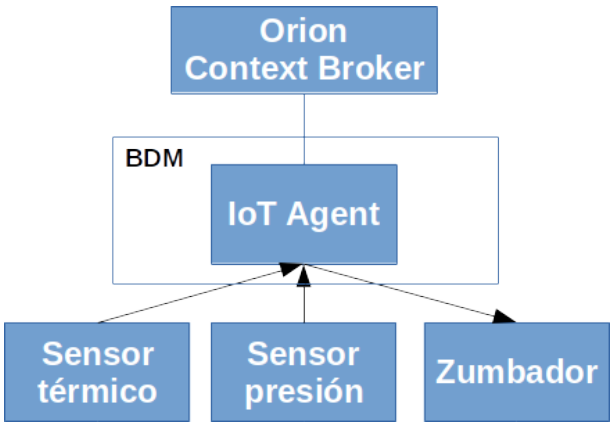


Figura 4.13: Ejemplo de comunicación entre CB y los dispositivos a través del IOTA.

Se ha decidido utilizar la implementación del IOTA Ultralight[13], libre de uso bajo la licencia GNU AFFERO GENERAL PUBLIC LICENSE v3[14]. Este software permite la comunicación con los protocolos AMQP, HTTP y MQTT, de modo que será posible enviar información con una petición HTTP POST y el IOTA se encargará de traducirla a NGSI.

Al tratarse de un software que suele ejecutarse en sistemas de bajo consumo o en los que la comunicación por red pueda ser un problema, el IOTA incorpora un sistema de mapeo que permite asignar alias a los parámetros recibidos para posteriormente mapearlos a la petición NGSI que enviará al CB. De este modo, es posible reducir el tamaño de los paquetes enviados acortando los nombres de los atributos a enviar. No obstante, es siempre necesario enviar la

[13]Enlace al repositorio de código del IoT Agent UL: `https://github.com/telefonicaid/iotagent-ul`
[14]Términos y condiciones de la licencia GNU AFFERO GENERAL PUBLIC LICENSE v3: `https://opensource.org/licenses/AGPL-3.0`

información que identifica al dispositivo realizando la petición y una clave sólo conocida por el dispositivo y por el IOTA para garantizar la seguridad.

#### IoT Edge

Compuesto por las herramientas que permiten conectar los dispositivos físicos a las aplicaciones FIWARE.

Tal como se expone en Gprivat (2018), las funcionalidades que el nivel de IoT Edge GEs implementa son:

- Adaptación de protocolo de dispositivos IoT. Funcionalidad reducida de la que proporciona el Backend Device Management Generic Enabler, aquí se limita a traducir protocolos específicos en NGSI para enviar directamente las entidades, normalmente a un IoT Broker o incluso al CB directamente. El Protocol Adapter GE es el encargado de proporcionar estas funciones.
- Procesamiento complejo de eventos. A pesar de existir un GE con este propósito específico, puede ser adecuado emplearlo en este nivel para reducir la cantidad de tráfico de red producido con la conexión con el CB. Funcionalidad proporcionada por el Data Handling GE generalmente desde un dispositivo que hace de puerta de acceso.
- Lógica de puerta de entrada. Ofrece una API que sirve como puente al dispositivo y para configurarlo. Estas funciones se implementan en el Device Management GE.
- Configuración de los nodos IoT. Algunos dispositivos capaces de conectarse a la red de forma autónoma no necesitarán puertas de acceso, por lo que podrán ser configurados y monitorizados de forma directa, por tanto, ofrecerán la API de configuración a nivel de dispositivo.
- Configuración de las redes IoT. En escenarios complejos no se espera que los dispositivos se conecten a una única red interconectada, por tanto, se ofrecen las herramientas para conectar los nodos a través de, por ejemplo, redes móviles (2G, 3G y 4G) o redes de radio. Esta funcionalidad es implementada por el Backend Device Management Generic Enabler, puertas de acceso o desde el propio nodo.

El sistema que se pretende construir no necesitará dispositivos que precisen de funcionalidades de este nivel de GEs. De este modo se concluye el análisis de los GEs de las categorías de administración de datos y contexto e IoT. En la siguiente sección se determina la forma de integrar los programas de detección de dispositivos explicados en la Sección 4.1: Wi-Fi y la Sección 4.2: Bluetooth,

## 4.4. Integración con detectores de dispositivos

La detección de los dispositivos a través de las señales Wi-Fi y Bluetooth se ha explicado con anterioridad, gracias a los programas desarrollados tras el análisis de las tecnologías, ha sido posible la adaptación de los datos generados al ecosistema FIWARE.

Para hacer posible el envío de los datos al CB, se utiliza el IOTA UL, una implementación que permite el envío de información desde los sensores al CB haciendo una traducción de protocolos.

El protocolo que acepta el programa elegido es Ultralight 2.0, se trata de un formato basado en texto dirigido a dispositivos cuya memoria y ancho de banda reservado a la comunicación son limitados. El formato utilizado para el envío de información es principalmente texto, separando las etiquetas y los valores con barras verticales (ver bloque de código 4.7).

```
t|15|k|abc
```

Código 4.7: Ejemplo de mensaje Ultralight 2.0 (fuente FIWARE IOT Agent UL).

Este protocolo también permite enviar varios mensajes a la vez, por lo que es posible acumular datos leídos y enviar en bloque para así reducir los tiempos de transmisión por red y, por ende, el consumo energético de los sensores.

Para iniciar el proceso de detección de dispositivos, se necesita configurar correctamente el entorno de sensorización, y ello conlleva crear los servicios básicos en el IOTA y registrar los dispositivos. Toda esta configuración se hace mediante el uso del puerto norte (Northbound port) que es el que recibiría los comandos del CB en caso de tener acciones registradas.

La creación de un servicio se realiza haciendo una petición HTTP al IOTA en la que se especifica, la ruta del servicio, el tipo por defecto de las entidades que manejará, la dirección del CB a la que enviar los datos recibidos y una contraseña que será utilizada para validar el acceso de los dispositivos al servicio (ver bloque de código 4.8).

```
curl -iX POST \
  'http://localhost:4041/iot/services' \
  -H 'Content-Type: application/json' \
  -H 'fiware-service: openiot' \
  -H 'fiware-servicepath: /' \
  -d '{
 "services": [
   {
     "apikey":      "4jggokgpepnvsb2uv4s40d59ov",
     "cbroker":     "http://orion:1026",
     "entity_type": "Thing",
     "resource":    "/iot/d"
   }
 ]
}'
```

Código 4.8: Creación de un servicio en el IOTA.

Una vez registrado el servicio, es necesario registrar los sensores que van a conectarse a él, de modo que se hace otra petición HTTP en la que se proporcionan los detalles del dispositivo. En este punto se establecen las correlaciones entre atributos representados por letras y su respectivo nombre a la hora de registrar ese valor en el CB, también se proporciona el tipo, el identificador y el nombre del dispositivo como entidad en el sistema (ver bloque de código

4.9).

```
curl -iX POST \
  'http://localhost:4041/iot/devices' \
  -H 'Content-Type: application/json' \
  -H 'fiware-service: openiot' \
  -H 'fiware-servicepath: /' \
  -d '{
 "devices": [
   {
     "device_id":   "sniffer001",
     "entity_name": "urn:ngsi:Sniffer:001",
     "entity_type": "Sniffer",
     "timezone":    "Europe/Berlin",
     "attributes": [
       { "object_id": "m", "name": "measure", "type": "String" }
     ]
   }
 ]
}'
```

Código 4.9: Registro de una entidad en el IOTA.

Al proceso en el que se realizan las acciones para la correcta configuración de un dispositivo se las conoce como proceso de aprovisionamiento de un dispositivo y, una vez finalizado, puede comprobarse si ha sido correctamente ejecutado realizando las peticiones correspondientes al IOTA sobre los recursos `services` (ver bloque de código 4.10) y `devices` (ver bloque de código 4.11).

```
// GET http://iot-agent.localhost/iot/services?resource=/iot/d&apikey
   =4jggokgpepnvsb2uv4s40d59ov
{
    "count": 1,
    "services": [
        {
            "commands": [],
            "lazy": [],
            "attributes": [],
            "_id": "5d73fc6807b0ee00114c96d2",
            "resource": "/iot/d",
            "apikey": "4jggokgpepnvsb2uv4s40d59ov",
            "service": "openiot",
            "subservice": "/",
            "__v": 0,
            "static_attributes": [],
            "internal_attributes": [],
            "entity_type": "Thing"
        }
    ]
```

```
}
```

Código 4.10: Listado de servicios creados en el IOTA.

```
// GET http://iot-agent.localhost/iot/devices/sniffer003
{
    "device_id": "sniffer001",
    "service": "openiot",
    "service_path": "/",
    "entity_name": "urn:ngsi:Sniffer:001",
    "entity_type": "Sniffer",
    "transport": "HTTP",
    "attributes": [
        {
            "object_id": "m",
            "name": "measure",
            "type": "String"
        }
    ],
    "lazy": [],
    "commands": [],
    "static_attributes": []
}
```

Código 4.11: Listado de entidades registradas en el IOTA.

Finalmente, para comprobar que el dispositivo ha sido correctamente registrado en el CB, y observar los valores iniciales de los atributos se puede hacer una petición a Orion indicando el identificador de la entidad que se ha registrado (ver bloque de código 4.12).

```
// GET http://orion.localhost/v2/entities/urn:ngsi:Sniffer:003?options
   =keyValues
{
    "id": "urn:ngsi:Sniffer:003",
    "type": "Sniffer",
    "TimeInstant": "2019-09-07T18:53:42.00Z",
    "measure": " "
}
```

Código 4.12: Atributos de la entidad creada en el CB.

Una vez el IOTA está preparado para recibir medidas de datos que serán propagadas al sistema. El registro de nuevas medidas se realiza a través del puerto sur del IOTA (Southbound port),y la forma de registrarlas es, al igual que para la configuración, a través de una petición HTTP en la que se envía la información haciendo uso de la correlación configurada previamente. En el bloque de código 4.13 se puede observar la petición ejecutada para registrar una detección de un dispositivo Wi-Fi.

```
curl -iX POST \
```

```
  'http://localhost:7896/iot/d?k=4jggokgpepnvsb2uv4s40d59ov&i=
     sniffer001' \
  -H 'Content-Type: text/plain' \
  -d 'm|bluetooth-ble
     f976cbb7e8c33bab2e5f68a66a2f23c490a7f77f6c515053ce9e52d53066fb37
      2019-08-24T12:31:09+00:00'
```

Código 4.13: Registro de valores para los atributos de una entidad en el IOTA.

Tras registrar una medición de valores en el IOTA, este propaga estos datos al CB y queda registrada para su consulta a través de la API (ver bloque de código 4.14).

```
// GET http://orion.localhost/v2/entities/urn:ngsi:Sniffer:003?options
   =keyValues
{
    "id": "urn:ngsi:Sniffer:001",
    "type": "Sniffer",
    "measure": "bluetooth-ble
       f976cbb7e8c33bab2e5f68a66a2f23c490a7f77f6c515053ce9e52d53066fb37
        2019-08-24T12:31:09+00:00"
},
```

Código 4.14: Entidad con atributos actualizados con la nueva información de contexto.

Una vez analizada la forma en la que registrar los sensores en el CB, tan solo se deben modificar los programas de detección de Wi-Fi y Bluetooth para que, en lugar de imprimir la información por pantalla, hagan una petición HTTP al IOTA UL con los datos en el formato especificado y los datos de autenticación definidos en la creación de los servicios. El código implementado para la detección de los dispositivos ha sido publicado en este repositorio con licencia MIT `https://github.com/peinad0/bluetooth-wifi-sniffer`.

Para agilizar las tareas de registro de sensores, se ha desarrollado un pequeño ejecutable que se encarga de hacer las peticiones necesarias para dejar las configuraciones y entidades creadas y preparadas para enviar datos al CB.

Tras la automatización del registro de los sensores en el sistema, se concluye el apartado de FIWARE. En el siguiente capítulo se muestra el estado final del sistema una vez integrados los dispositivos detectores de individuos así como los diferentes tipos de almacenamiento para la persistencia de los datos generados.

# 5 Sistema de detección de desplazamientos

En este capítulo se expone el sistema obtenido tras la investigación de las diferentes tecnologías utilizadas, la planificación, el desarrollo y la puesta en marcha del sistema. Por tanto, se detalla la arquitectura final del sistema de detección de desplazamientos y de qué forma se han integrado todos los componentes haciendo así posible almacenar los datos.

## 5.1. Arquitectura del sistema

La arquitectura diseñada para el sistema de detección de desplazamientos (ver figura 5.1) consta de los siguientes elementos.

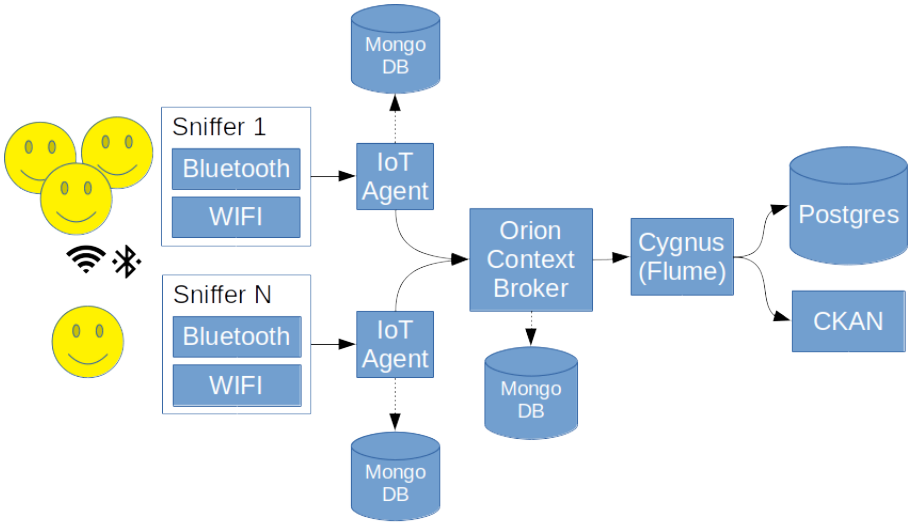


Figura 5.1: Arquitectura para el sistema de detección de desplazamientos.

- Sniffers o detectores de individuos. Se trata de dispositivos hardware que están ejecutando los programas de detección que detectan señales Wi-Fi y Bluetooth y registrando estas detecciones en el IOTA mediante peticiones HTTP.

- IoTAgent. Programa también integrado en el mismo dispositivo que los detectores y que realiza la traducción del protocolo UltraLight 2.0, utilizado por los sniffers, al protocolo NGSIv2 entendido por los componentes del ecosistema FIWARE. Estos agentes necesitan una base de dato para mantener las credenciales autorizadas a registrar datos y los servicios y dispositivos que hayan sido configurados.

- Orion Contex Broker. Encargado de mantener el registro de entidades del sistema así como sus atributos y valores. Necesita base de datos para poder mantener estos datos y las configuraciones de las suscripciones que permiten a otros GEs explotar la información para proporcionar un valor añadido.

- Cygnus. Recibe las actualizaciones de contexto realizadas en el CB mediante una suscripción a las entidades deseadas. Por cada actualización, se encarga de generar un evento para cada uno de los almacenamientos configurados.

- Postgres. Es el almacenamiento histórico del sistema, gracias a las inserciones realizadas por Cygnus, será posible mantener el registro de todos los cambios de contexto que han ocurrido sobre las entidades del sistema.

- CKAN. Plataforma de datos abiertos en la que se crea una organización y un set de datos sobre el que se vuelcan las actualizaciones de contexto en tiempo real.

## 5.2. Procesamiento de datos en el sistema

El almacenamiento de los datos en el sistema es una de las partes más importantes puesto que permite mantener el histórico de información de contexto para así aplicar técnicas de análisis y procesamiento de datos.

La persistencia, como anteriormente se ha explicado, es responsabilidad de Cygnus, que con las actualizaciones de las entidades a las que está suscrito, genera filas en una tabla que se crea automáticamente. Este GE crea una tabla por cada entidad sobre la que recibe actualizaciones con unas columnas fijas (ver figura 5.2).

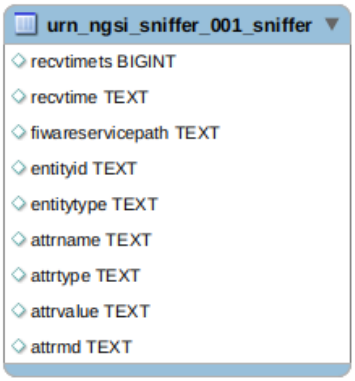


Figura 5.2: Tabla autogenerada por Cygnus para las entidades del sistema.

Así, por cada actualización de contexto de cada uno de los atributos de la entidad, Cygnus generará una nueva fila en la tabla. Esto supone una limitación, puesto que si se quisiera modelar una entidad con diferentes atributos, cada actualización conjunta de los tres atributos generaría tres filas en la tabla. Este es el motivo por el que se ha decidido modelar las entidades con un único atributo `measure` que contiene la información de un dispositivo detectado en un único campo formateado separando cada valor con un espacio (ver figura 5.3).

| | |
|---|---|
| **recvtimets** | 1567927048556 |
| **recvtime** | 2019-09-08T07:17:28.556Z |
| **fiwareservicepath** | / |
| **entityid** | urn:ngsi:Sniffer:001 |
| **entitytype** | Sniffer |
| **attrname** | measure |
| **attrtype** | String |
| **attrvalue** | bluetooth-ble<br>f976cbb7e8c33bab2e5f68a66a2f23c490a7f77f6c515053ce9e52d53066fb37<br>2019-08-24T12:31:09+00:00 |
| **attrmd** | [] |

Figura 5.3: Fila de la tabla de uno de los dispositivos con los datos de contexto.

La simple detección de dispositivos no es suficiente para conocer los desplazamientos de individuos entre los diferentes PITs sensorizados, es por esto que se hace necesario emplear técnicas de procesamiento de datos para poder generar información más sencilla de analizar.

Para generar datos fáciles de analizar se ha implementado un programa en Python que unifica los datos de todas las entidades registradas. Las acción principal que realiza es la lectura de todas las tablas correspondientes a entidades para volcar la información en una nueva tabla que contiene el identificador de la entidad que capturó la trama, el identificador del dispositivo capturado, la fuente de la que proviene la señal y el día y hora en el que se realizó la medición (ver figura 5.4a).


(a) Esquema de la tabla intermedia en la que se agregan los datos de las entidades.

(b) Esquema de la tabla final de desplazamientos.

Figura 5.4: Esquema de las tablas utilizadas en el procesamiento de datos de las entidades.

Con los datos obtenidos en esta tabla, ya es posible realizar representaciones gráficas de cantidad de individuos registrados en cada STD así como analizar qué posibles factores puedan afectar a la presencia de masas en estos puntos.

Sin embargo, teniendo como objetivo la detección de desplazamientos, se ha añadido un paso al programa en el que se identifican los dispositivos que se repiten entre las diferentes mediciones de distintas entidades y se ordenan cronológicamente, de este modo, cada fila de esta nueva tabla, representará un desplazamiento entre dos entidades, las cuales se encuentran localizadas geográficamente en diferentes puntos (ver figura 5.4b)

## 5.3. Simulación y caso de uso

En esta sección se muestra un caso de uso del sistema mediante una simulación de captura de datos y se explican los eventos que cada captura genera y de qué forma interpretar los datos tras ser almacenados en la base de datos de contexto histórico y procesados posteriormente por el programa descrito en el capítulo anterior.

Se presentan tres secciones en las que se planteará el escenario de la simulación, se mostrarán detalles de la ejecución y los datos y, finalmente, se hará uso estos mismos para generar mapas con información geolocalizada.

### 5.3.1. Escenario

Para la correcta simulación de este caso de uso, era necesario establecer la ubicación de los dispositivos detectores de individuos, para ello, se han elegido tres puntos con interés turístico en la ciudad de Torrevieja (ver figura 5.5).

- Ayuntamiento, donde se ha ubicado el dispositivo con identificador `sniffer001`.
- Laguna Rosa, donde se ha ubicado el dispositivo con identificador `sniffer002`.
- Paseo marítimo, donde se ha ubicado el dispositivo con identificador `sniffer003`.


Figura 5.5: Localización de los dispositivos con los sensores en la simulación.

Para la detección de individuos y conseguir un resultado semejante a un caso real, se va a utilizar un generador aleatorio de identificadores para crear datos que no se repitan y se inyectarán en el sistema cada cierto tiempo. Así se simulará que dispositivos no conocidos entran en el rango de detección y son registrados por el sistema. En cuanto a las detecciones reales, la prueba se realiza en un entorno controlado en el que no hay dispositivos emitiendo señales Wi-Fi o Bluetooth, tan solo un dispositivo que genera estas señales bajo demanda y, por tanto, sólo será detectado cuando sea requerido por el detector especificado. La información que se capturará de forma controlada puede consultarse en la tabla 5.1.

| Origen | Identificador | Dispositivo |
|---|---|---|
| wifi | 0b:ac:c4:88:98:c2 | sniffer001 |
| wifi | 0b:ac:c4:88:98:c2 | sniffer002 |
| wifi | 0b:ac:c4:88:98:c2 | sniffer003 |
| bluetooth | 19:52:86:4b:7a:f3 | sniffer001 |
| bluetooth | 19:52:86:4b:7a:f3 | sniffer002 |

Tabla 5.1: Datos de los dispositivos identificados de forma controlada en la simulación.

Los identificadores de dispositivos son modificados por el sistema con el fin de obtener cierto grado de privacidad, por lo que los valores que se verán registrados para el identificador de Wi-Fi y el de Bluetooth son los siguientes:

- Wi-Fi: `4d42473dfe5af611ca60b2243ecbdec94cc7f923358c3bd914c2762112aa8b7d`.
- Bluetooth: `3e7610df9fad8d8caa5f813ffa30edc25027be17e62a37ac26eb21157ec6fb9d`.

Los datos inyectados con esta información permitirán simular que se detectan tres desplazamientos de individuos y así se podrán mostrar algunas de las diferentes formas de representación que el sistema puede proporcionar.

### 5.3.2. Simulación

Una vez puesto en marcha el sistema, con los dispositivos correctamente configurados y registrados para poder enviar detecciones al CB, se procede a iniciar el programa generador de datos simulados.

El programa genera poco a poco datos y va actualizando el contexto de las entidades registradas. Durante este proceso, se añade la información controlada de forma que las actualizaciones de contexto controladas se mezclan con las simuladas. En el bloque de código 5.1 se puede apreciar la respuesta de una petición al CB para comprobar el estado de las entidades tras finalizar la simulación.

```
[
    {
        "id": "urn:ngsi:Sniffer:001",
        "type": "Sniffer",
        "measure": "wifi f685be4d800[...]0ea3b9ecdc77b3 2019-08-09T12
           :18:52+00:00"
    },
    {
        "id": "urn:ngsi:Sniffer:002",
        "type": "Sniffer",
        "measure": "wifi a83ee20af61[...]a97e98650f97b9 2019-08-09T18
           :18:30+00:00"
    },
    {
        "id": "urn:ngsi:Sniffer:003",
        "type": "Sniffer",
```

```
        "measure": "bluetooth 8ee56db1204[...]88ea8877fd6ac6
           2019-08-09T23:35:17+00:00"
    }
]
```

Código 5.1: Información de contexto de las entidades registradas.

Cada actualización de contexto de la simulación genera una fila en la base de datos, en la figura 5.6 se muestran diez mediciones registradas en el detector con nombre `sniffer001`.

| | entityid text | attrname text | attrvalue text |
|---|---|---|---|
| 1 | urn:ngsi:Sniffer:001 | measure | bluetooth-ble f8c962d8acaab2187ce4f30e74f2d732e19cd0c7ed13c94cb93c1b4395feb8ac 2019-07-31T07:12:02+00:00 |
| 2 | urn:ngsi:Sniffer:001 | measure | wifi c1eb52046a369c02cf99e5de1f17c3f45652528fa29f26fc5645460a6a7d3f62 2019-08-04T18:33:12+00:00 |
| 3 | urn:ngsi:Sniffer:001 | measure | bluetooth-ble 7a04926e6c9826674991b727335e1b55af668bcf6dce8b94305df5bcef3810aa 2019-08-04T16:44:26+00:00 |
| 4 | urn:ngsi:Sniffer:001 | measure | bluetooth-ble 9acac66e769aab18c2e22e1a601f59988fe1ed20b2ba21276542e69e08f639f2 2019-08-04T11:17:00+00:00 |
| 5 | urn:ngsi:Sniffer:001 | measure | bluetooth ccd4712359774ebf898de1ca873217eff34bbd7860bd6a8d63e165e3db09da18 2019-08-04T13:15:36+00:00 |
| 6 | urn:ngsi:Sniffer:001 | measure | wifi 3a2c26135022e2718b6737e9dc73c012e574410c552ae697bf177dc060b7fc87 2019-08-05T18:38:08+00:00 |
| 7 | urn:ngsi:Sniffer:001 | measure | bluetooth f04f482809be04208a2fe76e762628acf134c96d0de769f969b668cf77f66709 2019-08-03T18:19:35+00:00 |
| 8 | urn:ngsi:Sniffer:001 | measure | wifi 8af0a261c5737587c32fc8088d2806c7d4dbe68817b3559c037056db45c48613 2019-08-03T09:15:35+00:00 |
| 9 | urn:ngsi:Sniffer:001 | measure | wifi 19343712af57dbb0f105cb13f3fa64fa647b704fc4443ad602ee8ee62e2fc6f7 2019-08-02T23:18:59+00:00 |
| 10 | urn:ngsi:Sniffer:001 | measure | bluetooth 975b61e5651c35c29919272afdb7a9c761bc44fa6d0589e807be7f15621306ca 2019-08-07T13:24:15+00:00 |

Figura 5.6: Porción de los dispositivos capturados en la simulación del entorno.

Una vez terminado, se procede a comprobar que las actualizaciones controladas han sido correctamente registradas, para ello, se ejecuta el programa de procesamiento que crea la tabla `agregated` a partir de los datos presentes en las tablas de las entidades. Esta tabla contiene la información de cada una de las entidades procesada, por lo que es posible hacer consultas buscando un identificador en concreto (ver bloque de código 5.2).

```
-- Consulta dispositivo Wi-Fi
SELECT entityid, source, device_id, instant
FROM openiot.agregated
WHERE device_id='4
   d42473dfe5af611ca60b2243ecbdec94cc7f923358c3bd914c2762112aa8b7d'
ORDER BY entityid;

-- Consulta dispositivo Bluetooth
SELECT entityid, source, device_id, instant
FROM openiot.agregated
WHERE device_id='3
   e7610df9fad8d8caa5f813ffa30edc25027be17e62a37ac26eb21157ec6fb9d'
ORDER BY entityid;
```

Código 5.2: Consulta de actualizaciones de contexto para un identificador.

En las figuras 5.7 y 5.8 puede apreciarse la información generada en la tabla de datos agregados.

Esta será la tabla sobre la que se hagan transformaciones para crear los datos necesarios para las representaciones de la información en mapas.

| | entityId text | source text | device_id text | instant text |
|---|---|---|---|---|
| 1 | urn:ngsi:Sniffer:001 | wifi | 4d42473dfe5af611ca60b2243ecbdec94cc7f923358c3bd914c2762112aa8b7d | 2019-08-04T18:33:12+00:00 |
| 2 | urn:ngsi:Sniffer:002 | wifi | 4d42473dfe5af611ca60b2243ecbdec94cc7f923358c3bd914c2762112aa8b7d | 2019-08-04T10:29:42+00:00 |
| 3 | urn:ngsi:Sniffer:003 | wifi | 4d42473dfe5af611ca60b2243ecbdec94cc7f923358c3bd914c2762112aa8b7d | 2019-08-04T08:43:34+00:00 |

Figura 5.7: Información de contexto registrada para el dispositivo Wi-Fi.

| | entityId text | source text | device_id text | instant text |
|---|---|---|---|---|
| 1 | urn:ngsi:Sniffer:001 | bluetooth | 3e7610df9fad8d8caa5f813ffa30edc25027be17e62a37ac26eb21157ec6fb9d | 2019-08-04T13:15:36+00:00 |
| 2 | urn:ngsi:Sniffer:002 | bluetooth | 3e7610df9fad8d8caa5f813ffa30edc25027be17e62a37ac26eb21157ec6fb9d | 2019-08-04T19:32:45+00:00 |

Figura 5.8: Información de contexto registrada para el dispositivo Bluetooth.

### 5.3.3. Uso de los datos

La utilización de la información que genera el sistema puede realizarse de diferentes modos, no obstante, la forma más visual y sencilla de entender es hacerlo mediante mapas.

Para la visualización de la cantidad de individuos detectados en cada STD se ha elegido un mapa en el que los hexágonos varían de color menos intenso a más en función de la cantidad de dispositivos que registran. La figura 5.9 muestra este mapa construído con Carto [1] al que se le ha añadido un filtro por fecha.

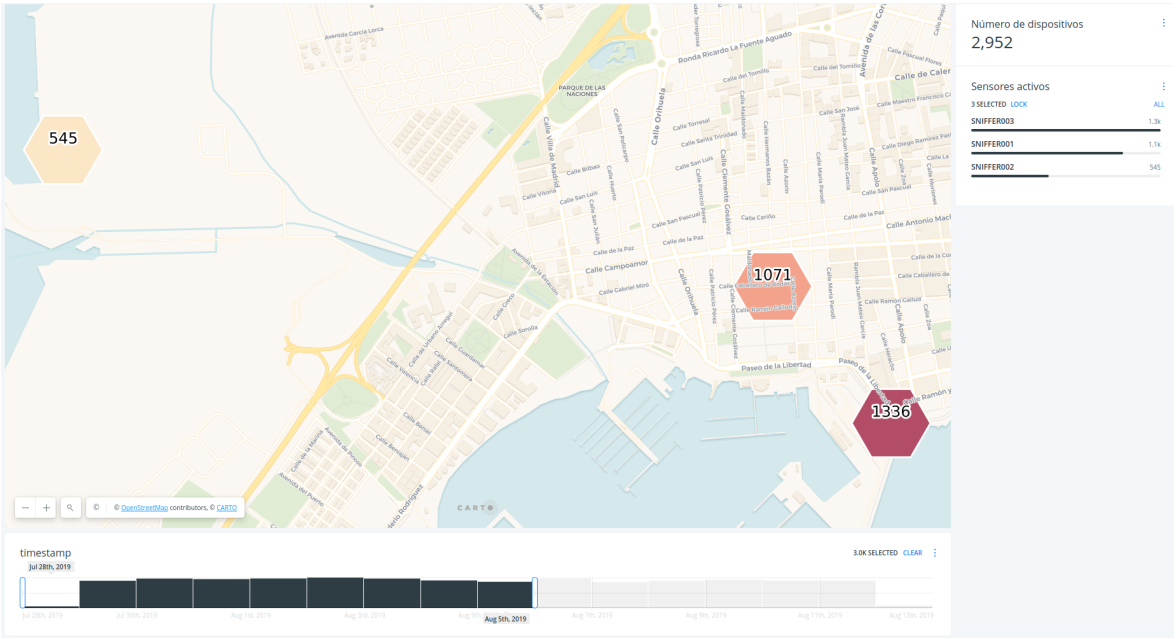


Figura 5.9: Mapa con numero de dispositivos detectados filtrados por fecha.

Gracias al filtro por fecha se puede depurar y analizar visualmente la cantidad de gente que hay en los puntos inteligentes situados en la ciudad en diferentes instantes de tiempo[2].

Los datos que se muestran en el mapa anterior son fruto de la simulación realizada, no obstante, se inyectaron datos reales para posteriormente poder detectar los desplazamientos.

[1] Enlace a constructor de mapas Carto: https://carto.com/

[2] Enlace a mapa interactivo de dispositivos detectados: `https://peinad0.carto.com/builder/3ada6df1-f544-4114-9c8c-5ac4c504cb16/embed`

En este caso, se ha elegido un mapa que muestra diferentes nodos, uno por cada dispositivo situado en la ciudad y que muestra, en base a uno de estos nodos, los desplazamientos que se han realizado dibujando una linea más o menos gruesa y de color más o menos intenso en función del número de individuos que ha realizado esta ruta.

En el mapa de la figura 5.10 puede observarse como, a partir del `sniffer001` se han dibujado los desplazamientos detectados.


Figura 5.10: Mapa desplazamientos desde dispositivo con nombre sniffer001.

Los movimientos de los individuos coinciden con los que se introdujeron de forma controlada durante la simulación (ver tabla 5.1). Dos dispositivos se han desplazado entre el `sniffer001` y el `sniffer002` y uno, que es el mismo que también realiza la ruta anterior, entre el `sniffer001` y el `sniffer003`. Así queda demostrado que la recolección de los datos y su posterior procesamiento funciona de forma adecuada y que los conjuntos de datos con información de contexto pueden ser utilizados para su representación sobre mapas.

Para finalizar el cuerpo del trabajo y dar paso a las conclusiones, cabe destacar que estas representaciones son tan solo dos formas de mostrar los datos, la parte más importante era la recolección y procesamiento de la información puesto que una vez obtenida, puede utilizarse de innumerables maneras

# 6 Conclusión

Para concluir el trabajo se hace retrospectiva y se comprueban los objetivos establecidos al principio para determinar si han sido cumplidos en mayor o menor medida.

Respecto al objetivo general marcado, desarrollar un sistema para la detección de desplazamiento de individuos aplicando FIWARE (O.G.1), se puede decir que ha sido cumplido puesto que el sistema diseñado e implementado permite detectar desplazamientos de dispositivos mediante sensores que envían los datos a través del ecosistema de FIWARE.

En cuanto a los objetivos específicos, el O.E.1 queda cumplido con el uso de librerías con licencias de código abierto, con el uso de FIWARE y con publicación de la implementación propia utilizando las herramientas mencionadas como código abierto[1].

La privacidad del usuario es un apartado a tener en cuenta en el funcionamiento del sistema, es por esto que se trata de anonimizar los datos de forma que la información almacenada nunca contiene el identificador del dispositivo sino una cadena generada a partir del mismo. Por tanto, se puede considerar el O.E.2 como cumplido.

En lo que respecta al objetivo O.E.3, no se ha implementado la publicación de datos en CKAN, la plataforma de código abierto, no obstante, se ha planteado el flujo necesario para realizar esta acción de modo que tan solo sería necesaria la configuración de Cygnus para este propósito, así que se considera el objetivo como parcialmente cumplido. Esto da paso al objetivo O.E.5, que queda conseguido gracias al uso de FIWARE, entre otros, se tiene el potencial de añadir cuantos componentes sean necesarios al sistema siempre que se comuniquen utilizando NGSIv2, además, el almacenamiento en una base de datos, permite incluso extender las funcionalidades del sistema a través de análisis de la información.

La toma de información y el procesamiento de los datos, O.E.4 se consigue también, en parte, por el uso de FIWARE. Esto se debe a que, por un lado, existe el flujo de actualizaciones de contexto que acaban siendo almacenadas en la base de datos y, por otro lado, se hace uso del programa que procesa los datos obteniendo una tabla agregada con la que poder realizar transformaciones para la representación de la información.

Por último, queda el objetivo O.E.6, que buscaba obtener un sistema fácil de poner en funcionamiento, esto se consigue gracias al uso de Docker. En una primera versión, el sistema se ejecuta sobre un único terminal, no obstante, la migración a servidores externos es mucho más sencilla ya que las dependencias quedan encapsuladas en las respectivas imágenes de los servicios.

En las siguientes secciones se habla de las limitaciones que tiene el sistema desarrollado así como posibles ampliaciones y líneas futuras tanto para la mejora funcional como para el aumento de la seguridad.

[1] Enlace a repositorio de código publicado: `https://github.com/peinad0/bluetooth-wifi-sniffer`

## 6.1. Limitaciones

La implementación del sistema de detección de desplazamientos tiene diversas limitaciones que hacen que los datos obtenidos no sean totalmente fiables. Entre las más importantes y que se han de tener en cuenta a la hora de la interpretación de los datos se encuentran:

- Aleatoriedad de MAC. Algunos dispositivos tienen implementado un sistema de cambio temporal de la MAC, esto implica que cada cierto tiempo se cambia la dirección MAC y, por tanto, si este cambio se produce durante el desplazamiento entre dos STD, el individuo no será identificado de igual modo, por lo que se perderá esta trazabilidad.
- No se capturan todos los paquetes. A pesar de utilizar métodos para buscar constantemente tramas Wi-Fi y Bluetooth, en ocasiones algunas se pierden debido a diferentes motivos, por lo que no se puede determinar que el número de dispositivos detectados sea el mismo que número de dispositivos que hayan entrado en el rango de detección de los sensores.
- Dispositivos múltiples por individuo. No es poco frecuente que un individuo porte varios dispositivos que utilicen la tecnología Wi-Fi o Bluetooth. Es por esto que las métricas pueden verse alteradas ya que se puede llegar a detectar un desplazamiento de un grupo de dispositivos y no tiene porqué significar el desplazamiento de ese número de individuos.
- Ética y legalidad. Leer información de paquetes enviados por diferentes frecuencias cuyo destinatario no es el propio dispositivo puede resultar una acción poco ética y estar cerca de infringir las leyes ya que el Código Penal sanciona el uso de tecnologías de la información para invadir o atentar contra la intimidad de las personas. En este caso no se pretende invadir esa intimidad y, además, se cumple el reglamento de la GDPR[2] puesto que la información que se almacena, teniendo en cuenta que los costes, el tiempo y la tecnología necesaria para la reversión del algoritmo SHA256, hacen que esa información tenga una probabilidad muy baja de ser procesada y utilizada para identificar a una persona.

Por último, se debe hacer un ejercicio de reflexión y tomar los datos proporcionados por el sistema como información meramente orientativa y que pueda servir de ayuda para la toma de decisiones de, por ejemplo, los departamentos turísticos de una ciudad.

## 6.2. Líneas futuras

El uso del ecosistema FIWARE da la posibilidad de extender el sistema con cualquier GE compatible con NGSIv2, es por esto que en cuanto a desarrollos futuros, existe una gran cantidad de opciones a tener en cuenta.

Para primar la seguridad, se propone introducir el GE KeyRock[3], un gestor de identidad que permite añadir OAuth2 como mecanismo de autenticación y autorización. Esto se considera

[2]Enlace a la normativa GDPR: `https://eugdpr.org/`
[3]Enlace a repositorio de código de KeyRock: `https://github.com/ging/fiware-idm`

un factor imprescindible para la puesta en producción del sistema de forma distribuida y segura.

Tal como se proponía en la sección en la que se explicaba Cygnus, el paso más sencillo y que puede aportar valor en forma de generación de conjuntos de datos abiertos es la conexión con CKAN. De este modo, se pueden publicar las actualizaciones de contexto y, al hacerlas públicas, se incentiva el desarrollo de aplicaciones basadas en estos datos, con lo que se genera una relación entre sistemas que puede ayudar a que ambos se mejoren y se adapten a las necesidades de uso.

También está la posibilidad de escalar el sistema horizontalmente, es decir, añadiendo más dispositivos para detectar individuos en un mayor número de emplazamientos y así aumentar las posibilidades de detección de desplazamientos. Además, con este aumento, la información climatológica o la incorporación de más sensores por dispositivo para generar detecciones más completas cobra importancia ya que permitirá realizar análisis de tendencias en las visitas a los diferentes puntos de la ciudad.

# Bibliografía


Becker, J. K., Li, D., y Starobinski, D. (2019). Tracking anonymized bluetooth devices. *Proceedings on Privacy Enhancing Technologies*, 2019(3):50–65.

Buhalis, D. y Amaranggana, A. (2013). Smart tourism destinations. In *Information and communication technologies in tourism 2014*, pages 553–564. Springer.

Buhalis, D. y Amaranggana, A. (2015). Smart tourism destinations enhancing tourism experience through personalisation of services. In Tussyadiah, I. y Inversini, A., editors, *Information and Communication Technologies in Tourism 2015*, pages 377–389, Cham. Springer International Publishing.

Committee, I. C. S. L. M. S. et al. (1999). Wireless lan medium access control (mac) and physical layer (phy) specifications. *ANSI/IEEE Std. 802.11-1999*.

Cunche, M. (2014). I know your mac address: Targeted tracking of individual using wi-fi. *Journal of Computer Virology and Hacking Techniques*, 10(4):219–227.

De Poorter, E., Moerman, I., y Demeester, P. (2011). Enabling direct connectivity between heterogeneous objects in the internet of things through a network-service-oriented architecture. *EURASIP Journal on Wireless Communications and Networking*, 2011(1):61.

Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7.

Donaire, J. A., Galí, N., et al. (2008). Modeling tourist itineraries in heritage cities. routes around the old district of girona. *COMITÉ EDITORIAL DIRECTOR: Agustín Santana Talavera*, 6:435.

E. Hall, R., Bowerman, B., Braverman, J., Taylor, J., Todosow, H., y Von Wimmersperg, U. (2000). The vision of a smart city. *2nd Int. Life .*

Erb, Y. (2011). Some aspects about the internet of things, the advantages and challenges: Business aspects of the internet of things. *Zurich: ETH Zurich*.

Federico M. Facca, J. F. (2018). How to use fiware harmonised data models in your projects. Disponible en `https://fiware-datamodels.readthedocs.io/en/latest/howto/index.html` [22/08/2019].

Fox, J. (2019). Fiware academy. Disponible en `https://fiware-academy.readthedocs.io/en/latest/` [20/08/2019].

Freudiger, J. (2015). How talkative is your mobile device?: an experimental study of wi-fi probe requests. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, page 8. ACM.

Gprivat, Mcp, T. R. (2018). Internet of things (iot) services enablement architecture. Disponible en `https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Internet_of_Things_(IoT)_Services_Enablement_Architecture` [22/08/2019].

Gubbi, J., Marusic, S., Rao, A. S., Law, Y. W., y Palaniswami, M. (2013). A pilot study of urban noise monitoring architecture using wireless sensor networks. In *2013 International conference on advances in computing, communications and informatics (ICACCI)*, pages 1047–1052. IEEE.

Gutiérrez, V., Galache, J. A., Sánchez, L., Muñoz, L., Hernández-Muñoz, J. M., Fernandes, J., y Presser, M. (2013). Smartsantander: Internet of things research and innovation through citizen participation. In Galis, A. y Gavras, A., editors, *The Future Internet*, pages 173–186, Berlin, Heidelberg. Springer Berlin Heidelberg.

Heydon, R. (2013). *Bluetooth low energy: the developer's handbook*. Prentice Hall.

Husted, N. y Myers, S. (2010). Mobile location tracking in metro areas: malnets and others. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 85–96. ACM.

Ivars-Baidal, J. A., Celdrán-Bernabeu, M. A., Mazón, J.-N., y Ángel F. Perles-Ivars (2019). Smart destinations and the evolution of icts: a new scenario for destination management? *Current Issues in Tourism*, 22(13):1581–1600.

Jameel, M. I. y Dungen, J. (2015). Low-power wireless advertising software library for distributed m2m and contextual iot. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 597–602. IEEE.

Javierlucio, M. (2016). Fiware architecture. Disponible en `https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_Architecture` [20/08/2019].

Jin, J., Gubbi, J., Marusic, S., y Palaniswami, M. (2014). An information framework for creating a smart city through internet of things. *IEEE Internet of Things journal*, 1(2):112–121.

Joostbr, Smg, S. (2016). Context management architecture. Disponible en `https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Data/Context_Management_Architecture` [20/08/2019].

Komninos, N. (2002). *Intelligent Cities*. Routledge.

Law, R., Rong, J., Vu, H. Q., Li, G., y Lee, H. A. (2011). Identifying changes and trends in hong kong outbound tourism. *Tourism Management*, 32(5):1106–1114.

Longo, E. (2017). Pairing wi-fi and bluetooth mac addresses through passive packets capture.

Mehmood, Y., Ahmad, F., Yaqoob, I., Adnane, A., Imran, M., y Guizani, S. (2017). Internet-of-things-based smart cities: Recent advances and challenges. *IEEE Communications Magazine*, 55(9):16–24.

Mei, Z., Wang, D., Chen, J., y Wang, W. (2014). Investigation of bicycle travel time estimation using bluetooth sensors for low sampling rates. *Promet-Traffic&Transportation*, 26(5):383–391.

Musa, A. y Eriksson, J. (2012). Tracking unmodified smartphones using wi-fi monitors. In *Proceedings of the 10th ACM conference on embedded network sensor systems*, pages 281–294. ACM.

Pei, L., Liu, J., Chen, Y., Chen, R., y Chen, L. (2017). Evaluation of fingerprinting-based wi-fi indoor localization coexisted with bluetooth. *The Journal of Global Positioning Systems*, 15(1):3.

Rose, M. D. (2017). Low-power wirelessly-linked rfid tracking system. US Patent 9,760,853.

SIG, B. (2001). *BLUETOOTH SPECIFICATION*. Bluetooth SIG. Version 1.1.

Stallings, W. (2005). Data and computer communications. *Prentice Hall*.

Vanhoef, M., Matte, C., Cunche, M., Cardoso, L. S., y Piessens, F. (2016). Why mac address randomization is not enough: An analysis of wi-fi network discovery mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '16, pages 413–424, New York, NY, USA. ACM.

Welch, D. y Lathrop, S. (2003). Wireless security threat taxonomy. In *IEEE Systems, Man and Cybernetics SocietyInformation Assurance Workshop, 2003.*, pages 76–83. IEEE.